C 8

```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII           VVV        EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII           VVV        EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII           VVV        EEEEEEEEEEEEEEE  RRR         RRR
```

C 16

```
**FILE**ID**RTTDRIVER
```

```
RRRRRRRR    TTTTTTTTTT  TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
RRRRRRRR    TTTTTTTTTT  TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
RR      RR      TT          TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
RR      RR      TT          TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
RR      RR      TT          TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
RR      RR      TT          TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
RRRRRRRR        TT          TT      DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
RRRRRRRR        TT          TT      DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
RR  RR          TT          TT      DD      DD  RR  RR          II      VV      VV  EE          RR  RR
RR  RR          TT          TT      DD      DD  RR  RR          II      VV      VV  EE          RR  RR
RR    RR        TT          TT      DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
RR    RR        TT          TT      DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
RR      RR      TT          TT      DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
RR      RR      TT          TT      DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....

LL              IIIIII        SSSSSSSS
LL              IIIIII        SSSSSSSS
LL                II        SS
LL                II        SS
LL                II        SS
LL                II        SS
LL                II          SSSSSS
LL                II          SSSSSS
LL                II                SS
LL                II                SS
LL                II                SS
LL                II                SS
LLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLL      IIIIII      SSSSSSSS
```

Table of contents

```
0000          .TITLE  RTTDRIVER - Remote Terminal Driver
0000          .IDENT  'V04-000'
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ;++
0000    ;
0000    ; FACILITY:
0000    ;
0000    ;       VAX/VMS Remote Terminal Driver
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;       This module contains the remote terminal driver routines.  This driver
0000    ;       is used by the application process side of the operation.  In other
0000    ;       words, it receives the QIO requests from the process that does not
0000    ;       have local access to the terminal.
0000    ;
0000    ;       This driver's primary function is to receive QIO system service
0000    ;       requests, repackage the QIO arguments, and hand the new package to
0000    ;       the tranport mechanism for delivery to the remote terminal
0000    ;       handler process on the system with local access to the terminal.
0000    ;       The transport mechanism is DECnet.  Netdriver is called directly
0000    ;       via the internal IRP mechanism.
0000    ;
0000    ; AUTHOR:
0000    ;
0000    ;       Len Kawell, 01-AUG-1979
0000    ;
0000    ; MODIFICATION HISTORY:
0000    ;
0000    ;       V03-014 JLV0390         Jake VanNoy             25-JUL-1984
0000    ;               Return ILLIOFUNC for FMS when PICSTRING is seen.
0000    ;
0000    ;       V03-013 LMP0275         L. Mark Pilant,         12-Jul-1984  12:42
0000    ;               Initialize the ACL info in the ORB to be a null descriptor
```

```
0000    ;               list rather than an empty queue.  This avoids the overhead
0000    ;               of locking and unlocking the ACL mutex, only to find out
0000    ;               that the ACL was empty.
0000    ;
0000    ;       V03-012 EMD0088         Ellen M. Dusseault      30-Apr-1984
0000    ;               Add DEV$M_NNM characteristic to DEVCHAR2 so that these
0000    ;               devices will have the "node$" prefix.
0000    ;
0000    ;       V03-011 LMP0221         L. Mark Pilant,         27-Mar-1984  11:53
0000    ;               Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
0000    ;               ORB$W_PROT.
0000    ;
0000    ;       V03-010 JLV0320         Jake VanNoy             18-DEC-1983
0000    ;               Remoe SS$_INCOMPAT from read fdt routine.  This error
0000    ;               is preventing set host from RSX and TOPS20.
0000    ;               Change write routine to send broadcast type message
0000    ;               if IO$M_BREAKTHRU is seen.  Remove RTT_BROADCAST routine
0000    ;               as it is obsolete.  Redo SET_MODE FDT to use case statement.
0000    ;               Clear io$m_extend bit in read routine. Remove CTRLC
0000    ;               and outband from SENSE_SPAWN.
0000    ;
0000    ;       V03-009 JLV0299         Jake VanNoy             30-JUL-1983
0000    ;               Add DEV$M_RTT to DPT_STORE's.
0000    ;
0000    ;       V03-008 JLV0252         Jake VanNoy             13-MAY-1983
0000    ;               Remove references to IO$M_ENABL_ALT and IO$M_DSABL_ALT.
0000    ;
0000    ;       V03-007 JLV0241         Jake VanNoy             20-APR-1983
0000    ;               Change ASSUME regarding TRM$_LASTITM.
0000    ;
0000    ;       V03-006 JLV0239         Jake VanNoy             29-MAR-1983
0000    ;               Add code to do new itemlist, remove V3.2 code to
0000    ;               handle read verify.
0000    ;
0000    ;       V03-005 JLV0227         Jake VanNoy             9-FEB-1983
0000    ;               Bug fix in error path of ALLOC_MESSAGE that caused
0000    ;               system crash. Another bug fix to the read fdt routine
0000    ;               that crashed system with large prompt size.
0000    ;
0000    ;       V03-004 JLV0215         Jake VanNoy             6-OCT-1982
0000    ;               Mods to SBL3007 to do parameter checking correctly.
0000    ;
0000    ;       V03-003 SBL3007         Steve Long      6-Aug-1982
0000    ;               Read verify support and permit IO$M_ENABL_ALT &
0000    ;               IO$M_DSABL_ALT to be processed in SETMODE
0000    ;
0000    ;       V03-002 DJD3007         Darrell Duffy   5-April-1982
0000    ;               Trap IO$M_ESCAPE and IO$M_EXTEND with reads to V2 systems.
0000    ;               Trap IO$M_ENABL_ALT IO$M_DSABL_ALT in SETMODE.
0000    ;
0000    ;       V03-001 DJD3006         Darrell Duffy   31-March-1982
0000    ;               Fix SENSEMODE TYPAHDCNT to return correct status.
0000    ;               Insert setting of mode bits for fixing spawn.
0000    ;
0000    ;       V02-016 DJD3005         Darrell Duffy   13-January-1982
0000    ;               Fix flushing of CTRL/Y to occur at deassign.
0000    ;               Use new cancel interface to distinguish cancel and deassign.
```

```
0000    115 ;
0000    116 ;       V02-015 DJD3004         Darrell Duffy   20-December-1981
0000    117 ;               Revert to use of attn ast processing for CTRL C and Y.
0000    118 ;               Remove privileges associated with declaring a ctrl/y ast.
0000    119 ;
0000    120 ;       V02-014 DJD3003         Darrell Duffy   24-November-1981
0000    121 ;               Add out-of-band ast support.  Fix bug in delivery
0000    122 ;               of hangup ast when the link has broken before it
0000    123 ;               was enabled.
0000    124 ;
0000    125 ;       V02-013 DJD3002         Darrell Duffy   12-November-1981
0000    126 ;               More of the same.
0000    127 ;
0000    128 ;       V02-012 DJD3001         Darrell Duffy   21-October-1981
0000    129 ;               Update for changes to terminal driver for V3.0
0000    130 ;
0000    131 ;       V02-011 DJD2004         Darrell Duffy   31-July-1981
0000    132 ;               Change broadcast interface to return failure on
0000    133 ;               terminal set for NOBROADCAST
0000    134 ;
0000    135 ;       V02-010 DJD2003         Darrell Duffy   2-May-1981
0000    136 ;               Fix double deallocate of rtt ucb.
0000    137 ;
0000    138 ;       V02-009 RLRLBCNT        Robert L. Rappaport     8-April-1981
0000    139 ;               Changes associated with IRP modifications to all BCNT
0000    140 ;               fields which have grown to longwords.  Also fix old bug
0000    141 ;               in RTT_WRITE which sometimes left garbage in R1.
0000    142 ;
0000    143 ;       V02-008 DJD2002         Darrell Duffy   8-Apr-1981
0000    144 ;               Fix race condition with broadcast messages after hangup.
0000    145 ;
0000    146 ;       V02-007 DJD2001         Darrell Duffy   5-Mar-1981
0000    147 ;               Change to call network driver directly to read and
0000    148 ;               write packets.
0000    149 ;
0000    150 ;       V02-006 LMK0006         Len Kawell      27-Feb-1981
0000    151 ;               Fix problem with immediate delivery of hangup AST when
0000    152 ;               AST is being cancelled.
0000    153 ;
0000    154 ;       1.05    LMK0005         Len Kawell      18-Mar-1980
0000    155 ;               Change broadcast to call EXE$ALONONPAGED.
0000    156 ;
0000    157 ;       1.04    LMK0004         Len Kawell      29-Feb-1980
0000    158 ;               Change adapter type in DPTAB to be NULL.
0000    159 ;
0000    160 ;       1.03    LMK0003         Len Kawell      25-Feb-1980
0000    161 ;               Change broadcast to not wait for completion to avoid
0000    162 ;               causing issuing process to indefinitely wait if delays
0000    163 ;               occur during remote delivery.
0000    164 ;
0000    165 ;       1.02    LMK0002         Len Kawell      21-Jan-1980
0000    166 ;               Add UCB$M_HANGUP flag so hangup is never lost.
0000    167 ;
0000    168 ;--
```

```
                   0000          .SBTTL  External and local symbol definitions
                   0000
                   0000   ;
                   0000   ; External symbols
                   0000   ;
                   0000
                   0000          $ACBDEF                         ; AST control block
                   0000          $AQBDEF                         ; ACP queue block
                   0000          $CANDEF                         ; Cancel interface codes
                   0000          $CRBDEF                         ; Channel request block
                   0000          $DCDEF                          ; Device classes and types
                   0000          $DDBDEF                         ; Device data block
                   0000          $DEVDEF                         ; Device characteristics
                   0000          $DYNDEF                         ; Buffer type codes
                   0000          $IDBDEF                         ; Interrupt data block
                   0000          $IODEF                          ; I/O function codes
                   0000          $IPLDEF                         ; Hardware IPL definitions
                   0000          $IRPDEF                         ; I/O request packet
                   0000          $JIBDEF                         ; Job Information block
                   0000          $MSGDEF                         ; Mailbox message types
                   0000          $ORBDEF                         ; OBJECT'S RIGHTS BLOCK OFFSETS
                   0000          $PCBDEF                         ; Process control block
                   0000          $PRDEF                          ; Processor registers
                   0000          $PRVDEF                         ; Privilege bits
                   0000          $PSLDEF                         ; Processor status longword
                   0000          $RBFDEF                         ; Remote Device Buffer definitions
                   0000          $RDPDEF                         ; Remote device packet
                   0000          $REMDEF                         ; General constants
                   0000          $SSDEF                          ; System status codes
                   0000          $TRMDEF                         ; Item list definitions
                   0000          $TTDEF                          ; Terminal definitions
                   0000          $TT2DEF                         ; More definitions
                   0000          $TTYDEF                         ; Terminal driver definitions
                   0000          $UCBDEF                         ; Unit control block
                   0000          $VCBDEF                         ; Volume control block
                   0000          $VECDEF                         ; Interrupt vector block
                   0000
                   0000   ;
                   0000   ; Local symbols
                   0000   ;
                   0000
                   0000   ;
                   0000   ; Argument list (AP) offsets for device-dependent QIO parameters
                   0000   ;
                   0000
         00000000  0000   P1     = 0                             ; First QIO parameter
         00000004  0000   P2     = 4                             ; Second QIO parameter
         00000008  0000   P3     = 8                             ; Third QIO parameter
         0000000C  0000   P4     = 12                            ; Fourth QIO parameter
         00000010  0000   P5     = 16                            ; Fifth QIO parameter
         00000014  0000   P6     = 20                            ; Sixth QIO parameter
                   0000
                   0000
                   0000   ;
                   0000   ; Other constants
                   0000   ;
                   0000
         00000008  0000   RTT$K_FIPL = 8                          ; IPL to synchronize
                   0000
                   0000   ;
                   0000   ; Definitions that follow the standard UCB fields
                   0000   ;
                   0000
                   0000           $RTTUCBEXT                      ; UCB Extensions
                   0000
         000000DE  0000   UCB$W_RTT_READERR = UCB$W_CT_QCTPCNT    ; unused cterm UCB field
                   0000
                   0000   ;
                   0000   ; Redefinitions of the irp fields
                   0000   ;
                   0000
         00000040  0000   IRP$W_RTT_COMPAT = IRP$Q_TT_STATE       ; Set for compatiblity error
                   0000
```

```
0000            .SBTTL  Standard tables
0000
0000    ;
0000    ; Driver prologue table
0000    ;
0000
0000            DPTAB   -                                       ; DPT-creation macro
0000                    END=RTT_END,-                           ; End of driver label
0000                    ADAPTER=NULL,-                          ; Adapter type
0000                    UCBSIZE=<UCB$K_RTT_LEN>,-               ; Length of UCB
0000                    NAME=RTTDRIVER                          ; Driver name
0038            DPT_STORE INIT                                  ; Start of load
0038                                                            ; initialization table
0038            DPT_STORE DDB,DDB$L_ACPD,L,<^A\REM\>            ; Default ACP name
003F            DPT_STORE DDB,DDB$L_ACPD+3,B,3                  ; ACP class
0043            DPT_STORE UCB,UCB$B_FIPL,B,RTT$K_FIPL           ; Device fork IPL
0047            DPT_STORE UCB,UCB$B_DIPL,B,RTT$K_FIPL           ; Device interrupt IPL
004B            DPT_STORE UCB,UCB$L_DEVCHAR,L,<-                ; Device characteristics
004B                    DEV$M_REC!-                             ;   record device
004B                    DEV$M_AVL!-                             ;   available
004B                    DEV$M_IDV!-                             ;   input device
004B                    DEV$M_ODV!-                             ;   output device
004B                    DEV$M_TRM!-                             ;   terminal device
004B                    DEV$M_CCL>                              ;   carriage control device
0052            DPT_STORE UCB,UCB$L_DEVCHAR2,L,<-               ; Device characteristics
0052                    DEV$M_RTT!-                             ; remote terminal UCB extension
0052                    DEV$M_NNM>                              ; prefix with "node$"
0059            DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_TERM         ; Terminal device
005D            DPT_STORE UCB,UCB$B_DEVTYPE,B,TT$_UNKNOWN       ; Unknown type
0061            DPT_STORE UCB,UCB$W_DEVBUFSIZ,@W,TTY$GW_DEFBUF ; Default buffer size
0068            DPT_STORE UCB,UCB$L_DEVDEPEND,@L,TTY$GL_DEFCHAR ; Default characteristics
006F            DPT_STORE ORB,ORB$B_FLAGS,B,-                   ; Protection block flags
006F                            <ORB$M_PROT_16>                 ; SOGW protection word
0073            DPT_STORE ORB,ORB$W_PROT,@W,TTY$GW_PROT         ; Default allocation protection
007A            DPT_STORE ORB,ORB$L_OWNER,@L,TTY$GL_OWNUIC ; Default owner UIC
0081
0081            DPT_STORE REINIT                                ; Start of reload
0081                                                            ; initialization table
0081            DPT_STORE DDB,DDB$L_DDT,D,RTT$DDT               ; Address of DDT
0086            DPT_STORE CRB,CRB$L_INTD+4,D,-                  ; Address of interrupt
0086                    RTT_INTERRUPT                           ; service routine
008B
008B            DPT_STORE END                                   ; End of initialization
0000                                                            ; tables
0000
0000    ;
0000    ; Driver dispatch table
0000    ;
0000
0000            DDTAB   -                                       ; DDT-creation macro
0000                    DEVNAM=RTT,-                            ; Name of device
0000                    FUNCTB=RTT_FUNCTABLE,-                  ; FDT address
0000                    UNSOLIC=RTT_UNSOLIC,-                   ; Unsolicited attention routine
0000                    CANCEL=RTT_CANCEL                       ; Cancel I/O routine
0038
0038    ;
0038    ; Function dispatch table
```

```
0038     ;
0038
0038     RTT_FUNCTABLE:                                          ; FDT for driver
0038                     FUNCTAB ,-                              ; Valid I/O functions
0038                             <READVBLK,-                     ; Read virtual
0038                             READLBLK,-                      ; Read logical
0038                             READPBLK,-                      ; Read physical
0038                             READPROMPT,-                    ; Read with prompt
0038                             TTYREADALL,-                    ; Read passall
0038                             TTYREADPALL,-                   ; Read with prompt passall
0038                             WRITEVBLK,-                     ; Write virtual
0038                             WRITELBLK,-                     ; Write logical
0038                             WRITEPBLK,-                     ; Write physical
0038                             SENSEMODE,-                     ; Sense device mode
0038                             SENSECHAR,-                     ; Sense device characteristics
0038                             SETMODE,-                       ; Set device mode
0038                             SETCHAR>                        ; Set device characteristics
0040                     FUNCTAB ,-                              ; Buffered functions
0040                             <READVBLK,-                     ; Read virtual
0040                             READLBLK,-                      ; Read logical
0040                             READPBLK,-                      ; Read physical
0040                             READPROMPT,-                    ; Read with prompt
0040                             TTYREADALL,-                    ; Read passall
0040                             TTYREADPALL,-                   ; Read with prompt passall
0040                             WRITEVBLK,-                     ; Write virtual
0040                             WRITELBLK,-                     ; Write logical
0040                             WRITEPBLK,-                     ; Write physical
0040                             SENSEMODE,-                     ; Sense device mode
0040                             SENSECHAR,-                     ; Sense device characteristics
0040                             SETMODE,-                       ; Set device mode
0040                             SETCHAR>                        ; Set device characteristics
0048                     FUNCTAB RTT_READ,-                      ; FDT read routine for
0048                             <READVBLK,-                     ; read virtual,
0048                             READLBLK,-                      ; read logical,
0048                             READPBLK,-                      ; read physical,
0048                             READPROMPT,-                    ; read with prompt
0048                             TTYREADALL,-                    ; read passall,
0048                             TTYREADPALL>                    ; and read with prompt passall
0054                     FUNCTAB RTT_WRITE,-                     ; FDT write routine for
0054                             <WRITEVBLK,-                    ; write virtual,
0054                             WRITELBLK,-                     ; write logical,
0054                             WRITEPBLK>                      ; and write physical.
0060                     FUNCTAB RTT_SENSEMODE,-                 ; FDT sense mode routine
0060                             <SENSECHAR,-                    ; for sense characteristics
0060                             SENSEMODE>                      ; and sense mode.
006C                     FUNCTAB RTT_SETMODE,-                   ; FDT set mode routine
006C                             <SETCHAR,-                      ; for set characteristics and
006C                             SETMODE>                        ; set mode.
```

```
                        0078          .SBTTL  RTT_WRITE - Function Decision Routine for WRITE Functions
                        0078  ;++
                        0078  ; RTT_WRITE - Function Decision Routine for WRITE Functions
                        0078  ;
                        0078  ; Functional description:
                        0078  ;
                        0078  ;       This routine is called by the SYS$QIO service to dispatch a WRITE
                        0078  ;       I/O request.
                        0078  ;
                        0078  ;       The QIO parameters for terminal WRITES are:
                        0078  ;
                        0078  ;       P1 = address of the buffer
                        0078  ;       P2 = size of the buffer
                        0078  ;       P3 = (unused)
                        0078  ;       P4 = carriage control specifier
                        0078  ;
                        0078  ;       The buffer is validated for access, the process's quota checked and
                        0078  ;       decremented, the data and carriage control are copied to a message
                        0078  ;       block, the address of the message block is stored in the IRP,
                        0078  ;       and the IRP is queued to the ACP for delivery to the remote system.
                        0078  ;
                        0078  ; Inputs:
                        0078  ;
                        0078  ;       R0-R2 = scratch registers
                        0078  ;       R3 = address of the IRP (I/O request packet)
                        0078  ;       R4 = address of the PCB (process control block)
                        0078  ;       R5 = address of the UCB (unit control block)
                        0078  ;       R6 = address of the CCB (channel control block)
                        0078  ;       R7 = bit number of the I/O function code
                        0078  ;       R8 = address of the FDT table entry for this routine
                        0078  ;       R9-R11 = scratch registers
                        0078  ;       AP = address of the 1st function dependent QIO parameter
                        0078  ;
                        0078  ; Outputs:
                        0078  ;
                        0078  ;       IRP$L_SVAPTE(R3) = address of message buffer
                        0078  ;       IRP$W_BOFF(R3) = size of message buffer
                        0078  ;       IRP$W_BCNT(R3) = size of user buffer
                        0078  ;
                        0078  ;       The routine preserves all registers except R0-R2, and
                        0078  ;       R9-R11.
                        0078  ;
                        0078  ;--
                        0078  RTT_WRITE:                              ; WRITE FDT routine
      56    6C    D0    0078          MOVL    P1(AP),R6               ; Get user buffer virtual address
      50    56    D0    007B          MOVL    R6,R0                   ; Set up for write check call
   57    04 AC    3C    007E          MOVZWL  P2(AP),R7               ; Get buffer size
      51    57    D0    0082          MOVL    R7,R1                   ; Set up for write check call
            06    13    0085          BEQL    10$                     ; Skip check if zero
  00000000'GF     16    0087          JSB     G^EXE$WRITECHK          ; Check buffer access
                        008D                                          ; (no return means no access)
                        008D  ;
                        008D  ; Allocate the message buffer
                        008D  ;
                        008D  10$:
      51    20    C0    008D          ADDL    #RBF$T_TT_WDATA,R1      ; Add header to request size
        03EB      30    0090          BSBW    ALLOC_MESSAGE           ; Allocate the message buffer
```

```
                           0093  ;
                           0093  ; Copy the data and carriage control to the message
                           0093  ;
     18 A2    57     3C    0093          MOVZWL  R7,RBF$L_TT_BCNT(R2)    ; Set requested byte count
              3C     BB    0097          PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
   54    20 A3       3C    0099          MOVZWL  IRP$W_FUNC(R3),R4       ; save function code and modifiers
   5A    0C AC       D0    009D          MOVL    P4(AP),R10              ; save carriage control
   09 54      09     E1    00A1          BBC     #IO$V_BREAKTHRU,R4,20$  ; Branch if not breakthru
                           00A5  ;
                           00A5  ; Format message so that it looks like the old broadcast message. Note
                           00A5  ; carriage control is cleared. This is a shortcoming
                           00A5  ; in this implementation, but this code will be obsolete shortly...
                           00A5  ;
     0E A2    01     AE    00A5          MNEGW   #1,RBF$W_OPCODE(R2)     ; Set function code for broadcast
         10 A2       B4    00A9          CLRW    RBF$W_MOD(R2)           ; No modifier bits here
              5A     D4    00AC          CLRL    R10                     ; set no carriage control
                           00AE  20$:
 20 A2    66  57     28    00AE          MOVC3   R7,(R6),RBF$T_TT_WDATA(R2) ; Copy data
        51    53     D0    00B3          MOVL    R3,R1                   ; Save adr beyond data
              3C     BA    00B6          POPR    #^M<R2,R3,R4,R5>        ; Restore the registers
     1C A2    5A     D0    00B8          MOVL    R10,RBF$L_TT_CARCON(R2) ; Copy carriage control
                           00BC  ;
                           00BC  ; Send the message to the remote device and exit QIO service
                           00BC  ;
        52    51     D0    00BC          MOVL    R1,R2                   ; Pointer beyond data in message
         40 A3       B4    00BF          CLRW    IRP$W_RTT_COMPAT(R3)    ; No compatibility error
          06B6       31    00C2          BRW     RTT_NETMSGSENDX         ;
```

```
                   00C5              .SBTTL  RTT_READ - Function Decision Routine for READ Functions
                   00C5
                   00C5      ;++
                   00C5      ; RTT_READ - Function Decision Routine for READ Functions
                   00C5      ;
                   00C5      ; Functional description:
                   00C5      ;
                   00C5      ;       This routine is called by the SYS$QIO service to dispatch a READ
                   00C5      ;       I/O request.
                   00C5      ;
                   00C5      ;       The QIO parameters for terminal READS are:
                   00C5      ;
                   00C5      ;       P1 = address of the buffer
                   00C5      ;       P2 = size of the buffer
                   00C5      ;       P3 = number of seconds to wait for characters
                   00C5      ;       P4 = address of terminator class bitmask or 0 if standard
                   00C5      ;       P5 = address of prompt string for IO$_READPROMPT or IO$_TTYREADPALL
                   00C5      ;       P6 = size of prompt string for IO$_READPROMPT or IO$_TTYREADPALL
                   00C5      ;
                   00C5      ;       The buffer is validated for access, the process's quota checked and
                   00C5      ;       decremented, the timeout, terminator mask, and prompt are copied to a
                   00C5      ;       message block, the address of the message block is stored in the IRP,
                   00C5      ;       and the IRP is queued to the ACP for delivery to the remote system.
                   00C5      ;
                   00C5      ; Inputs:
                   00C5      ;
                   00C5      ;       R0-R2 = scratch registers
                   00C5      ;       R3 = address of the IRP (I/O request packet)
                   00C5      ;       R4 = address of the PCB (process control block)
                   00C5      ;       R5 = address of the UCB (unit control block)
                   00C5      ;       R6 = address of the CCB (channel control block)
                   00C5      ;       R7 = bit number of the I/O function code
                   00C5      ;       R8 = address of the FDT table entry for this routine
                   00C5      ;       R9-R11 = scratch registers
                   00C5      ;       AP = address of the 1st function dependent QIO parameter
                   00C5      ;
                   00C5      ; Outputs:
                   00C5      ;
                   00C5      ;       IRP$L_SVAPTE(R3) = address of message buffer
                   00C5      ;       IRP$W_BOFF(R3) = size of message buffer
                   00C5      ;       IRP$L_MEDIA(R3) = address of user buffer
                   00C5      ;       IRP$W_BCNT(R3) = size of user buffer
                   00C5      ;
                   00C5      ;       The routine preserves all registers except R0-R2, and
                   00C5      ;       R9-R11.
                   00C5      ;
                   00C5      ;--
                   00C5
                   00C5      ; Local storage offsets on stack:
                   00C5
00000000           00C5      bufaddr = 0
00000004           00C5      bufsize = 4
00000008           00C5      prmaddr = 8
0000000C           00C5      prmsize = 12
00000010           00C5      trmaddr = 16
00000014           00C5      trmsize = 20
00000018           00C5      iniaddr = 24
```

```
              0000001C  00C5  inisize = 28
              00000020  00C5  timeout = 32
              00000024  00C5  inioffset = 36
                        00C5
              00000028  00C5  read_local = 40
                        00C5
                        00C5  RTT_READ:                                      ; READ FDT routine
                        00C5  ;
                        00C5  ; Set up stack locals
                        00C5  ;
      5E    28      C2  00C5          SUBL2   #READ_LOCAL,SP         ; Allocate local storage
      58    5E      D0  00C8          MOVL    SP,R8                  ; Save pointer
            68      7C  00CB          CLRQ    (R8)                   ; clear buf ***
         08 A8      7C  00CD          CLRQ    8(R8)                  ; clear prm ...
         10 A8      7C  00D0          CLRQ    16(R8)                 ; clear trm ...
         18 A8      7C  00D3          CLRQ    24(R8)                 ; clear ini ...
         20 A8      7C  00D6          CLRQ    32(R8)                 ; clear other ...
                        00D9  ;
                        00D9  ; Check access to user's buffer
                        00D9  ;
      50    6C      D0  00D9          MOVL    P1(AP),R0              ; Get user buffer virtual address
   38 A3    50      D0  00DC          MOVL    R0,IRP$L_MEDIA(R3)     ; Save address in packet
   51    04 AC      3C  00E0          MOVZWL  P2(AP),R1              ; Get buffer size
            09      13  00E4          BEQL    10$                    ; Skip check if zero
      68    50      7D  00E6          MOVQ    R0,BUFADDR(R8)         ; Set up for read check call
   00000000'GF      16  00E9          JSB     G^EXE$READCHK          ; Check buffer access
                        00EF                                         ; (no return means no access)
                        00EF  ;
                        00EF  ; Check for extended itemlist read
                        00EF  ;
                        00EF  10$:
            0F      E5  00EF          BBCC    #IO$V_EXTEND,-         ; If this is not item list
      06 20 A3          00F1                  IRP$W_FUNC(R3),15$     ; then continue
          00A9      30  00F4          BSBW    RT_READ_ITMLST         ; process item list
          0059      31  00F7          BRW     200$                   ; continue
                        00FA  ;
                        00FA  ; Get prompt, if specified
                        00FA  ;
                        00FA  15$:
      37    57      91  00FA          CMPB    R7,#IO$_READPROMPT     ; Read prompt?
            05      13  00FD          BEQL    20$                    ; Branch if yes
      3B    57      91  00FF          CMPB    R7,#IO$_TTYREADPALL    ; Read prompt?
            14      12  0102          BNEQ    50$                    ; Branch if not
   51    14 AC      3C  0104  20$:    MOVZWL  P6(AP),R1              ; Get size of prompt
            0E      13  0108          BEQL    50$                    ; If eql then make this normal read
   50    10 AC      D0  010A          MOVL    P5(AP),R0              ; Get prompt buffer address
                        010E  ;
                        010E  ; Check access to prompt string
                        010E  ;
   08 A8    50      7D  010E          MOVQ    R0,PRMADDR(R8)         ; Save address and size
   00000000'GF      16  0112          JSB     G^EXE$WRITECHK         ; Check prompt access
                        0118  ;
                        0118  ; Get terminator bitmask and check access
                        0118  ;
                        0118  50$:
            52      D4  0118          CLRL    R2                     ; Assume no terminator specified
   51    0C AC      D0  011A          MOVL    P4(AP),R1              ; Get address of terminator desc
```

```
             2A    13  011E              BEQL    65$                     ; If eql none specified
     50      0C    3C  0120              MOVZWL  #SS$_ACCVIO,R0          ; Assume no access
                       0123              IFNORD  #8,(R1),63$             ; Descriptor accessible?
     52      61    3C  0129              MOVZWL  (R1),R2                 ; Get bitmask size
             08    12  012C              BNEQ    60$                     ; If neq long format
     52      04    D0  012E              MOVL    #4,R2                   ; Size of short format
     51      04    C0  0131              ADDL    #4,R1                   ; Set address of bitmask
             14    11  0134              BRB     65$                     ;
                       0136     60$:
  51      04 A1    D0  0136              MOVL    4(R1),R1                ; Get address of long format bitmask
                       013A              IFNORD  R2,(R1),63$             ; Bitmask accessible?
     20      52    B1  0140              CMPW    R2,#32                  ; Bitmask greater than allowed size?
             05    1B  0143              BLEQU   65$                     ; If gtru yes
     50      14    3C  0145              MOVZWL  #SS$_BADPARAM,R0        ; bad parameter
             50    11  0148     63$:     BRB     READ_ERROR
                       014A     65$:
   10 A8     51    7D  014A              MOVQ    R1,TRMADDR(R8)          ; terminator address and size
20 A8     08 AC    D0  014E              MOVL    P3(AP),TIMEOUT(R8)      ; Set timeout value
                       0153     200$:
                       0153     ;
                       0153     ; Common code again, Allocate the message buffer
                       0153     ;
  5B      04 A8    D0  0153              MOVL    BUFSIZE(R8),R11         ; Set size of read
  32 A3      5B    B0  0157              MOVW    R11,IRP$W_BCNT(R3)      ; Reset read buffer size
                       015B                                              ; (modified by EXE$WRITECHK)
                       015B
     51      23    D0  015B              MOVL    #RBF$T_TT_TERM+3,R1     ; Set header + overhead size
  51      0C A8    C0  015E              ADDL    PRMSIZE(R8),R1          ; Prompt size
  51      14 A8    C0  0162              ADDL    TRMSIZE(R8),R1          ; terminator size
           0315    30  0166              BSBW    ALLOC_MESSAGE           ; Allocate the message buffer
                       0169     ;
                       0169     ; Copy the timeout, terminator bitmask, and prompt string to the message
                       0169     ;
  18 A2      5B    D0  0169              MOVL    R11,RBF$L_TT_BCNT(R2)   ; Set requested byte count
          20 A8    D0  016D              MOVL    TIMEOUT(R8),-
          1C A2        0170                      RBF$L_TT_TIMOUT(R2)     ; Set timeout value
             3C    BB  0172              PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
                       0174
  50      10 A8    7D  0174              MOVQ    TRMADDR(R8),R0          ; Set terminator addr and size
  20 A2      51    90  0178              MOVB    R1,RBF$T_TT_TERM(R2)    ; Set terminator bitmask size
21 A2   60   51    28  017C              MOVC    R1,(R0),RBF$T_TT_TERM+1(R2) ; Copy terminator bitmask
                       0181
  50      08 A8    7D  0181              MOVQ    PRMADDR(R8),R0          ; Set prompt addr and size
     83      51    B0  0185              MOVW    R1,(R3)+                ; Set size of prompt
  63    60   51    28  0188              MOVC    R1,(R0),(R3)            ; Copy prompt string
                       018C
     51      53    D0  018C              MOVL    R3,R1                   ; Save adr beyond data
             3C    BA  018F              POPR    #^M<R2,R3,R4,R5>        ; Restore registers
                       0191     ;
                       0191     ; Send the message the remote device and exit the QIO service
                       0191     ;
     52      51    D0  0191              MOVL    R1,R2                   ; Set address beyond data
          40 A3    B4  0194              CLRW    IRP$W_RTT_COMPAT(R3)    ; No compatiblity error
           05E1    31  0197              BRW     RTT_NETMSGSENDX         ;
                       019A     ;
                       019A     ; Error in processing
                       019A     ;
```

```
                           019A   607 READ_ERROR:                               ; READ FDT error
        00000000'GF   17   019A   608          JMP     G^EXE$ABORTIO             ; Abort the I/O request
```

```
                              01A0          .SBTTL  RT_READ_ITMLST - FDT routine for read with item list
                              01A0  ;++
                              01A0  ;
                              01A0  ;
                              01A0  ;
                              01A0  ; *** a clean up pass is needed to here to verify that the paranoia
                              01A0  ;     checks made by TTDRIVER and this driver are the same.
                              01A0  ;
                              01A0  ;--
                              01A0
                              01A0  RT_READ_ITMLST:
                              01A0
                              01A0          ;
                              01A0          ; Set up probe of itemlist with P3 as access mode
                              01A0          ;
               56    53   D0  01A0          MOVL    R3,R6                   ; Save IRP
    50   08 AC   02    00   EF  01A3          EXTZV   #0,#2,P3(AP),R0         ; fetch low 2 bits of parameter
          00000000'GF    16   01A9          JSB     G^EXE$MAXACMODE         ; maximize with mode of caller
               53    50   D0  01AF          MOVL    R0,R3                   ; Set input to probe routine
                              01B2
            50   10 AC    D0  01B2          MOVL    P5(AP),R0               ; Address of itemlist
            51   14 AC    D0  01B6          MOVL    P6(AP),R1               ; size of item list
                     05   13  01BA          BEQL    10$                     ; can't be zero?
               5A    50   7D  01BC          MOVQ    R0,R10                  ; save both
                     08   11  01BF          BRB     30$                     ; ok, continue
               50    14   3C  01C1  10$:    MOVZWL  #SS$_BADPARAM,R0        ; status
               53    56   D0  01C4  20$:    MOVL    R6,R3                   ; Restore IRP
                     D1   11  01C7          BRB     READ_ERROR              ; abort
                              01C9
          00000000'GF    16   01C9  30$:    JSB     G^EXE$PROBER            ; Can it be read?
                  F2 50   E9  01CF          BLBC    R0,20$                  ; branch if not
               50    5B   D0  01D2          MOVL    R11,R0                  ; size
                              01D5          ;
                              01D5          ; Verify that size is multiple of 12
                              01D5          ;
               53    56   D0  01D5          MOVL    R6,R3                   ; Restore IRP
                     51   D4  01D8          CLRL    R1                      ; quadword r0/r1
    50   5B    50    0C   7B  01DA          EDIV    #12,R0,R11,R0           ; divide
                     50   D5  01DF          TSTL    R0                      ; must be zero remainder
                     DE   12  01E1          BNEQ    10$                     ; error
                              01E3          ;
                              01E3          ; Now loop and conquer item list, item by item
                              01E3          ;
                              01E3  40$:
               51    8A   3C  01E3          MOVZWL  (R10)+,R1               ; Length
               52    8A   3C  01E6          MOVZWL  (R10)+,R2               ; item code
               50    8A   D0  01E9          MOVL    (R10)+,R0               ; address or immediate value
                     8A   D5  01EC          TSTL    (R10)+                  ; Must be zero field
                     D1   12  01EE          BNEQ    10$                     ; error if not
                              01F0
                              01F0          CASE    R2,-                    ; case on message type
                              01F0                  <100$,-                 ; TRM$_MODIFIERS        (0)
                              01F0                  200$,-                  ; TRM$_EDITMODE         (1)
                              01F0                  300$,-                  ; TRM$_TIMEOUT          (2)
                              01F0                  400$,-                  ; TRM$_TERM             (3)
                              01F0                  500$,-                  ; TRM$_PROMPT           (4)
                              01F0                  600$,-                  ; TRM$_INISTRING        (5)
```

```
                        01F0              700$,-                   ; TRM$_PICSTRING        (6)
                        01F0              800$,-                   ; TRM$_FILLCHR          (7)
                        01F0              900$,-                   ; TRM$_INIOFFSET        (8)
                        01F0              1000$-                   ; TRM$_ALTECHSTR        (9)
                        01F0              >,-                      ; TRM$_LASTITM          (10)
                        01F0              TYPE = W
                        0208
                        0208      ASSUME  TRM$_LASTITM EQ 10       ; Break assembly if not right
              B7    11  0208      BRB     10$
                        020A
                        020A 100$:                 ; TRM$_MODIFIERS
                        020A
   50   8000 8F     AA  020A      BICW    #IO$M_EXTEND,R0          ; clear extend bit
      20 A3  50     A8  020F      BISW    R0,IRP$W_FUNC(R3)        ; Set read flags
             5A     11  0213      BRB     2000$                    ; Loop
                        0215
                        0215 200$:                 ; TRM$_EDITMODE
             58     11  0215      BRB     2000$                    ; ignore
                        0217
                        0217 300$:                 ; TRM$_TIMEOUT
                        0217
      20 A8  50     D0  0217      MOVL    R0,TIMEOUT(R8)           ; Set timeout
 20 A3  0080 8F     A8  021B      BISW    #IO$M_TIMED,IRP$W_FUNC(R3) ; set read timed bit
             4C     11  0221      BRB     2000$                    ; loop
                        0223
                        0223 400$:                 ; TRM$_TERM
             51     D5  0223      TSTL    R1                       ; test length
             09     12  0225      BNEQ    410$                     ; If neq long format
         51  04     D0  0227      MOVL    #4,R1                    ; Size of short format
   50    F8  AA     9E  022A      MOVAB   -8(R10),R0               ; Address of immediate data *** hack
             13     11  022E      BRB     430$                     ; skip
                        0230 410$:
                        0230      IFNORD  R1,(R0),420$             ; Bitmask accessible?
         20  51     B1  0236      CMPW    R1,#32                   ; Bitmask greater than allowed size?
             08     1B  0239      BLEQU   430$                     ; If less than or equal, no
             84     11  023B      BRB     10$                      ; bad param *** other status?
         50  0C     3C  023D 420$: MOVZWL #SS$_ACCVIO,R0           ; access violation
           FF57     31  0240      BRW     READ_ERROR               ; branch to read error
                        0243 430$:
      10 A8  50     7D  0243      MOVQ    R0,TRMADDR(R8)           ; save address and size of terminators
             26     11  0247      BRB     2000$                    ; continue
                        0249
                        0249 500$:                 ; TRM$_PROMPT
      08 A8  50     7D  0249      MOVQ    R0,PRMADDR(R8)           ; save address and length
             37     F0  024D      INSV    #IO$_READPROMPT,-
         06  00         024F              #IRP$V_FCODE,#IRP$S_FCODE,-
          20 A3         0251              IRP$W_FUNC(R3)           ; Set Read with prompt
             0C     11  0253      BRB     650$                     ; continue
                        0255
                        0255 700$:                 ; TRM$_PICSTRING
   50   00F4 8F     3C  0255      MOVZWL  #SS$_ILLIOFUNC,R0        ; for FMS...
           FF3D     31  025A      BRW     READ_ERROR
                        025D
                        025D 1000$:                ; TRM$_ALTECOSTR
                        025D 600$:                 ; TRM$_INISTRING
      18 A8  50     7D  025D      MOVQ    R0,INIADDR(R8)           ; save address and length
             51     D5  0261 650$: TSTL   R1                       ; no need to check if zero
```

```
                 0A   13  0263                  BEQL    2000$                       ; Skip parameter
                 0F   10  0265                  BSBB    CHK_READERR                 ; check for read error
                 06   11  0267                  BRB     2000$                       ; continue
                          0269
                          0269      800$:                           ; TRM$_FILLCHR
                          0269      900$:                           ; TRM$_INIOFFSET
                 50   B5  0269                  TSTW    R0                          ; test to see if present
                 02   13  026B                  BEQL    2000$                       ; branch if not
                 07   10  026D                  BSBB    CHK_READERR                 ; check for read error
                          026F
                          026F      2000$:
              01 5B   F5  026F                  SOBGTR  R11,2010$                   ; loop
                      05  0272                  RSB
                          0273
               FF6D   31  0273      2010$:      BRW     40$                         ;
                          0276
                          0276      CHK_READERR:
                          0276
     50   00DE C5   3C  0276                    MOVZWL  UCB$W_RTT_READERR(R5),R0 ; set status
                 01   B0  027B                  MOVW    #SS$_NORMAL,-
            00DE C5       027D                          UCB$W_RTT_READERR(R5)   ; set success if this happens again
              01 50   E9  0280                  BLBC    R0,10$                      ; branch if error
                      05  0283                  RSB                                 ; continue without error
               FF13   31  0284      10$:        BRW     READ_ERROR                  ; abort
                          0287
```

```
0287            .SBTTL  RTT_SETMODE, Function Decision Routine for SETMODE/SETCHAR
0287    ;++
0287    ; RTT_SETMODE, Function Decision Routine for SETMODE/SETCHAR Functions
0287    ;
0287    ; Functional description:
0287    ;
0287    ;       This routine is called by the SYS$QIO service to dispatch a SETMODE
0287    ;       or SETCHAR I/O request.
0287    ;
0287    ;       The QIO parameters for terminal SETMODE or SETCHAR are:
0287    ;
0287    ;
0287    ;               P1 = address of 8 byte characteristics buffer
0287    ;               P2 = 0, 8 or 12
0287    ;               P3 = speed specifier
0287    ;               P4 = fill specifier
0287    ;               P5 = parity flags
0287    ;
0287    ;       IO$V_CTRLYAST -
0287    ;               P1 = AST routine address or zero to cancel
0287    ;
0287    ;       IO$V_CTRLCAST -
0287    ;                P1 = AST routine address or zero to cancel
0287    ;
0287    ;       IO$V_HANGUP -
0287    ;               NONE
0287    ;
0287    ;       The buffer (if any) is validated for access, the process's quota
0287    ;       checked and decremented, a message block is allocated, the parameters
0287    ;       (if any) are stored in the  message block, the address of the message
0287    ;       block is stored in the IRP, and the IRP is queued to the ACP for
0287    ;       delivery to the remote system.
0287    ;
0287    ;       If an AST is to be enabled, an AST control block is allocated locally
0287    ;       hung off the UCB for later delivery upon receipt of a corresponding
0287    ;       attention message from the remote system.
0287    ;
0287    ; Inputs:
0287    ;
0287    ;       R0-R2 = scratch registers
0287    ;       R3 = address of the IRP (I/O request packet)
0287    ;       R4 = address of the PCB (process control block)
0287    ;       R5 = address of the UCB (unit control block)
0287    ;       R6 = address of the CCB (channel control block)
0287    ;       R7 = bit number of the I/O function code
0287    ;       R8 = address of the FDT table entry for this routine
0287    ;       R9-R11 = scratch registers
0287    ;       AP = address of the 1st function dependent QIO parameter
0287    ;
0287    ; Outputs:
0287    ;
0287    ;       IRP$L_SVAPTE(R3) = address of message buffer
0287    ;       IRP$W_BOFF(R3) = size of message buffer
0287    ;
0287    ;       The routine preserves all registers except R0-R2, R7, and R9-R11
0287    ;--
0287    RTT_SETMODE:                                    ; SETMODE/SETCHAR FDT routine
```

```
           40 A3    B4  0287           CLRW    IRP$W_RTT_COMPAT(R3)    ; No compatibility error
        50 20 A3    3C  028A           MOVZWL  IRP$W_FUNC(R3),R0       ; Fetch function code and modifers
  51    50 09 06    EA  028E           FFS     #IO$V_MAINT,#9,R0,R1    ; Find first set modifier
              33    13  0293           BEQL    SET_CHAR                ; if none then simple set mode.
                        0295
     50   0380 8F   B3  0295           BITW    #<IO$M_CTRLCAST!-
                        029A                   IO$M_CTRLYAST!-
                        029A                   IO$M_HANGUP>,R0         ; Always legal functions
              0E    12  029A           BNEQ    30$                     ; branch if any of these
                        029C
         00D5 C5    95  029C           TSTB    UCB$B_RTT_PROECO(R5)    ; Previous version
              08    12  02A0           BNEQ    30$                     ; Nope
     50   069C 8F   3C  02A2           MOVZWL  #SS$_INCOMPAT+3, R0     ; Abort maintenance, outband, etc.
            010F    31  02A7           BRW     ABORT                   ; with an error not success
                        02AA   30$:
                        02AA           CASE    R1,TYPE=B,LIMIT=#IO$V_MAINT,<-
                        02AA                   SET_MAINT,-             ; IO$M_MAINT
                        02AA                   SET_CTRLY,-             ; IO$M_CTRLYAST
                        02AA                   SET_CTRLC,-             ; IO$M_CTRLCAST
                        02AA                   SET_HANGUP,-            ; IO$M_HANGUP
                        02AA                   SET_OUTBAND,-           ; IO$M_OUTBAND
                        02AA                   SET_CONNECT,-           ; IO$M_CONNECT
                        02AA                   SET_DISCONNECT,-        ; IO$M_DISCONNECT
                        02AA                   SET_PID,-               ; IO$M_SETPID
                        02AA                   SET_BRDCST>             ; IO$M_BRDCST
                        02C0   ;
                        02C0   ; invalid characteristic if CASE falls though
                        02C0   ;
     50   00F4 8F   3C  02C0           MOVZWL  #SS$_ILLIOFUNC, R0      ; Return as illegal operation
            00F1    31  02C5           BRW     ABORT                   ; with an error not success
                        02C8
                        02C8   SET_CHAR:
            00FD    30  02C8           BSBW    GET_PARAMS              ; validate and fetch parameters
        5B 48 A5    D0  02CB           MOVL    UCB$L_DEVDEPND2(R5),R11 ; Extended word is defaulted
           59 81    7D  02CF           MOVQ    (R1)+,R9                ; Get characteristics
           0C 52    D1  02D2           CMPL    R2, #12                 ; Do we get another longword?
              03    19  02D5           BLSS    20$                     ; Nope
           5B 81    D0  02D7           MOVL    (R1)+, R11              ; Obtain the third longword
        40 A5 59    7D  02DA   20$:    MOVQ    R9,UCB$B_DEVCLASS(R5)   ; Set local copy of characteristics
        48 A5 5B    D0  02DE           MOVL    R11,UCB$L_DEVDEPND2(R5) ; And extended longword
                        02E2
         00D5 C5    95  02E2           TSTB    UCB$B_RTT_PROECO(R5)    ; If old version
              10    12  02E6           BNEQ    30$                     ; Nope
     00F00000 8F    D3  02E8           BITL    # <<<1@24>-1>-<<1@TT$V_HALFDUP>-1>>,-
           44 A5        02EE                   UCB$L_DEVDEPEND(R5)     ; If extra bits set, then
              06    13  02F0           BEQL    30$                     ; return incompat error
         0699 8F    B0  02F2           MOVW    #SS$_INCOMPAT,-         ; but carry on with function
           40 A3        02F6                   IRP$W_RTT_COMPAT(R3)    ;
                        02F8   30$:
            004F    31  02F8           BRW     SET_MESSAGE             ; send message
                        02FB           ;
                        02FB           ; The following types of modifiers are not allowed on remote terminals
                        02FB           ;
                        02FB   SET_MAINT:
                        02FB   SET_CONNECT:
                        02FB   SET_DISCONNECT:
                        02FB
```

```
          50   0334 8F   3C  02FB   864          MOVZWL  #SS$_DEVREQERR, R0        ; Return as device request error
                  00B6   31  0300   865          BRW     ABORT                     ; with an error not success
                             0303   866
                             0303   867 SET_BRDCST:
                  00C2   30  0303   868          BSBW    GET_PARAMS                ; Get parameters
           00A8 C5   61  7D  0306   869          MOVQ    (R1),UCB$Q_TL_BRKTHRU(R5); Set bits
                    06   11  030B   870          BRB     SET_NOP                   ; Set done
                             030D   871
                             030D   872 SET_PID:
                 60 A4   D0  030D   873          MOVL    PCB$L_PID(R4),-
               00A4 C5       0310   874                  UCB$L_TL_CTLPID(R5)       ; Set controlling PID
                             0313   875 SET_NOP:
                  00A9   31  0313   876          BRW     FDT_FINISHIOC_OK          ; Complete I/O
                             0316   877
                             0316   878 SET_CTRLY:
          57   0090 C5   DE  0316   879          MOVAL   UCB$L_RTT_CTRLY(R5),R7    ; Get address of CNTRL/Y AST list
           00000000'GF   16  031B   880          JSB     G^COM$SETATTNAST          ; Enable an attention AST
                    03   E1  0321   881          BBC     #UCB$V_TT_HANGUP,-
              21 68 A5       0323   882                  UCB$W_DEVSTS(R5),CTRL_CY ; Branch if no hangup
               50   67   D0  0326   883          MOVL    (R7),R0                   ; Get address of AST block
                    1C   13  0329   884          BEQL    CTRL_CY                   ; If eql, no AST to deliver
               54   57   D0  032B   885          MOVL    R7,R4                     ; Set address of AST listhead
     1C A0   02CC 8F     3C  032E   886          MOVZWL  #SS$_HANGUP,ACB$L_KAST+4(R0) ; Set AST parameter to hangup
           00000000'GF   16  0334   887          JSB     G^COM$DELATTNAST          ; Deliver the AST immediately
                    D7   11  033A   888          BRB     SET_NOP                   ; finish I/O
                             033C   889
                             033C   890 SET_CTRLC:
          57   0094 C5   DE  033C   891          MOVAL   UCB$L_RTT_CTRLC(R5),R7    ; set CNTRL/C AST enable
           00000000'GF   16  0341   892          JSB     G^COM$SETATTNAST          ; Enable an attention AST
                             0347   893
                             0347   894 CTRL_CY:
             59    6C    D0  0347   895          MOVL    P1(AP),R9                 ; Get address of AST routine
                             034A   896                                            ; fall htrough to send message
                             034A   897 ;
                             034A   898 ; Create SET message and send to remote device
                             034A   899 ;
                             034A   900 SET_HANGUP:
                             034A   901 SET_MESSAGE:                               ; Create and queue SET message
             51    30    D0  034A   902          MOVL    #RBF$L_TT_CHAR2+4,R1      ; Set size of message buffer
                  012E   30  034D   903          BSBW    ALLOC_MESSAGE             ; Allocate a message buffer
           18 A2   59    7D  0350   904          MOVQ    R9,RBF$Q_TT_CHAR(R2)      ; Set characteristics or AST parameter
           2C A2   5B    D0  0354   905          MOVL    R11, RBF$L_TT_CHAR2(R2)   ; And the next longword
     20 A2    08 AC      D0  0358   906          MOVL    P3(AP),RBF$L_TT_SPEED(R2) ; Set speed
     24 A2    0C AC      D0  035D   907          MOVL    P4(AP),RBF$L_TT_FILL(R2)  ; Set fills
     28 A2    10 AC      D0  0362   908          MOVL    P5(AP),RBF$L_TT_PARITY(R2) ; Set parity
       01      00D5 C5   91  0367   909          CMPB    UCB$B_RTT_PROECO(R5), -   ; How long should the message be?
                             036C   910                  #REM$C_CURECO             ; Long or short
                    06   12  036C   911          BNEQ    10$                       ; Shorter message
          52    30 A2    9E  036E   912          MOVAB   RBF$L_TT_CHAR2+4(R2),R2  ; Address of longer message
                    04   11  0372   913          BRB     20$
          52    2C A2    9E  0374   914 10$:     MOVAB   RBF$L_TT_PARITY+4(R2),R2; Set address beyond data
                  0400   31  0378   915 20$:     BRW     RTT_NETMSGSENDX           ; Send message to remote and exit service
                             037B   916
                             037B   917 ;
                             037B   918 ; Process a setmode for an outofband ast
                             037B   919 ;
                             037B   920
```

```
                              037B  SET_OUTBAND:
 0C 20 A3   0B   E0   037B                  BBS     #IO$V_INCLUDE, -          ; Include list?
                      0380                          IRP$W_FUNC(R3), 10$       ;
      009C C5   9E   0380                   MOVAB   UCB$L_RTT_BANDEXCL(R5),-  ; Address of exclude ast list
           57        0384                           R7                        ;
      0098 C5   9E   0385                   MOVAB   UCB$L_RTT_BANDEXMSK(R5),- ; Address of the exclude mask
           52        0389                           R2                        ;
           0A   11   038A                   BRB     20$
                     038C
      00C4 C5   9E   038C  10$:             MOVAB   UCB$L_RTT_BANDINCL(R5),-  ; Address of include ast list
           57        0390                           R7                        ;
      00C8 C5   9E   0391                   MOVAB   UCB$L_RTT_BANDINMSK(R5),- ; Address of the include mask
           52        0395                           R2                        ;
                     0396  20$
   00000000'GF  16   0396                   JSB     G^COM$SETCTRLAST          ; Enable the asts
           22   D0   039C                   MOVL    #RBF$B_TT_OUTBAND+1+4+1+4,-;
           51        039E                           R1                        ; Set size of message
         00DC   30   039F                   BSBW    ALLOC_MESSAGE             ; Allocate a message
        18 A2   9E   03A2                   MOVAB   RBF$B_TT_OUTBAND(R2),-    ; Address of data in message
           52        03A5                           R2                        ;
     82    04   90   03A6                   MOVB    #4, (R2)+                 ; Count for include mask
      00C8 C5   D0   03A9                   MOVL    UCB$L_RTT_BANDINMSK(R5),-; Include mask
           82        03AD                           (R2)+                     ;
     82    04   90   03AE                   MOVB    #4, (R2)+                 ; Count for exclude mask
      0098 C5   D0   03B1                   MOVL    UCB$L_RTT_BANDEXMSK(R5),- ; Now the exclude mask
           82        03B5                           (R2)+                     ;
         03C2   31   03B6                   BRW     RTT_NETMSGSENDX           ; Send the message
```

```
                        03B9          .SBTTL  ABORT, Transfer to EXE$ABORTIO
                        03B9
                        03B9  ;
                        03B9  ; Error processing - abort I/O request
                        03B9  ;
                        03B9  ABORT:
      00000000'GF   17  03B9          JMP     G^EXE$ABORTIO            ;
                        03BF
                        03BF  ;
                        03BF  ; Finish I/O, clear R1
                        03BF  ;
                        03BF  FDT_FINISHIOC_OK:
         50    01   3C  03BF          MOVZWL  #SS$_NORMAL,R0           ; Set status OK
                        03C2  FDT_FINISHIOC:
      00000000'GF   17  03C2          JMP     G^EXE$FINISHIOC          ; Complete I/O request
                        03C8
                        03C8          .SBTTL  GET_PARAMS - Get set mode parameters
```

```
                          03C8  ;++
                          03C8  ; GET_PARAMS
                          03C8  ;
                          03C8  ; inputs
                          03C8  ;        AP -> qio argument list
                          03C8  ;
                          03C8  ; outputs
                          03C8  ;        r1 = address of parameters
                          03C8  ;        r2 = 8 or 12 for size of characteristics buffer
                          03C8  ;
                          03C8  ; ABORT if P2(ap) is not 0, 8, 12.
                          03C8  ; Return ss$_incompat if not current system and size is 12.
                          03C8  ;--
                          03C8
                          03C8  GET_PARAMS:
                          03C8
          51    6C    D0  03C8          MOVL    P1(AP),R1                 ; Get address of characteristics
                0C    10  03CB          BSBB    RTT_CHARSIZE              ; Obtain the size of the char buffer
          50    0C    3C  03CD          MOVZWL  #SS$_ACCVIO,R0            ; Assume access violation
                          03D0          IFNORD  R2,(R1),10$               ; Characteristics accessible?
                      05  03D6          RSB                               ; return
                          03D7  10$:
                E0    11  03D7          BRB     ABORT                     ; error
                          03D9
                          03D9          .SBTTL  RTT_CHARSIZE, Size of characteristics buffer
                          03D9
                          03D9  RTT_CHARSIZE:
       52    04 AC    D0  03D9          MOVL    P2(AP), R2       ; Size of characters buffer
                0F    13  03DD          BEQL    10$              ; Zero is for 8
          08    52    D1  03DF          CMPL    R2, #8           ; 8 is allowed
                0D    13  03E2          BEQL    20$              ; Ok
                0C    1F  03E4          BLSSU   30$              ; Less is no good
                10    10  03E6          BSBB    RTT_ECOQ         ; If greater then we must be latest
          0C    52    D1  03E8          CMPL    R2, #12          ; Must be 12 and nothing else
                05    12  03EB          BNEQ    30$              ; No good
                      05  03ED          RSB                      ; Ok
          52    08    D0  03EE  10$:    MOVL    #8, R2           ; Use 8 if zero
                      05  03F1  20$:    RSB
                          03F2
          50    14    3C  03F2  30$:    MOVZWL  #SS$_BADPARAM, R0         ; Abort qio with an error
              FFC1    31  03F5          BRW     ABORT
                          03F8
                          03F8          .SBTTL  RTT_ECOQ, Validate latest eco number
                          03F8  ;++
                          03F8  ; RTT_ECOQ
                          03F8  ;
                          03F8  ; inputs
                          03F8  ;        r3 -> irp
                          03F8  ;        r5 -> ucb
                          03F8  ; outputs
                          03F8  ;        return if eco is latest,
                          03F8  ;        else abort QIO with ss$_badparam
                          03F8  ;--
                          03F8
                          03F8  RTT_ECOQ:
             40 A3    B4  03F8          CLRW    IRP$W_RTT_COMPAT(R3)      ; Make sure its zero
           00D5 C5    95  03FB          TSTB    UCB$B_RTT_PROECO(R5)      ; Latest for now is just a one
```

```
                        06  12  03FF   1024        BNEQ    10$                      ; zero is last eco level
                   0699 8F  B0  0401   1025        MOVW    #SS$_INCOMPAT,-          ; Return quiet error
                     40 A3      0405   1026                IRP$Q_RTT_COMPAT(R3)     ; message
                            05  0407   1027 10$:   RSB
```

```
                                  0408          .SBTTL  RTT_SENSEMODE, Function Decision Routine for SENSEMODE/SENSECHAR
                                  0408  ;++
                                  0408  ; RTT_SENSEMODE, Function Decision Routine for SENSEMODE/SENSECHAR Functions
                                  0408  ;
                                  0408  ; Functional description:
                                  0408  ;
                                  0408  ;       This routine is called by the SYS$QIO service to dispatch a SENSEMODE
                                  0408  ;       or SENSECHAR I/O request.
                                  0408  ;
                                  0408  ;       The QIO parameters for terminal SENSEMODE/SENSECHAR are:
                                  0408  ;
                                  0408  ;       P1 = address of 8 or 12 byte characteristics buffer
                                  0408  ;       P2 = 0, 8 or 12
                                  0408  ;
                                  0408  ;       The buffer is validated for access, the process's quota checked and
                                  0408  ;       decremented, a message block is allocated, the address of the message
                                  0408  ;       block is stored in the IRP, and the IRP is queued to the ACP for
                                  0408  ;       delivery to the remote system.
                                  0408  ;
                                  0408  ; Inputs:
                                  0408  ;
                                  0408  ;       R0-R2 = scratch registers
                                  0408  ;       R3 = address of the IRP (I/O request packet)
                                  0408  ;       R4 = address of the PCB (process control block)
                                  0408  ;       R5 = address of the UCB (unit control block)
                                  0408  ;       R6 = address of the CCB (channel control block)
                                  0408  ;       R7 = bit number of the I/O function code
                                  0408  ;       R8 = address of the FDT table entry for this routine
                                  0408  ;       R9-R11 = scratch registers
                                  0408  ;       AP = address of the 1st function dependent QIO parameter
                                  0408  ;
                                  0408  ; Outputs:
                                  0408  ;
                                  0408  ;       IRP$L_SVAPTE(R3) = address of message buffer
                                  0408  ;       IRP$W_BOFF(R3) = size of message buffer
                                  0408  ;       IRP$L_MEDIA(R3) = address of user characteristics buffer
                                  0408  ;       IRP$W_BCNT(R3) = size of user characteristics buffer, 8
                                  0408  ;
                                  0408  ;       The routine preserves all registers except R0-R2, and R9-R11
                                  0408  ;--
                                  0408  RTT_SENSEMODE:                                  ; SENSEMODE/SENSECHAR FDT routine
            40 A3      B4         0408          CLRW    IRP$W_RTT_COMPAT(R3)    ; No compatibility error
                                  040B
      59    20 A3      3C         040B          MOVZWL  IRP$W_FUNC(R3),R9       ; Fetch function code
   08 59    07         E1         040F          BBC     #IO$V_RD_MODEM,R9,5$    ; skip if not read modem
   50     0334 8F      3C         0413          MOVZWL  #SS$_DEVREQERR, R0      ; Return as device request error
            FF9E       31         0418          BRW     ABORT                   ; with an error not success
                                  041B  5$:
         51    6C      D0         041B          MOVL    P1(AP),R1               ; Get address of characteristics buffer
            FFB8       30         041E          BSBW    RTT_CHARSIZE            ; Size of chars buffer (return in R2)
         50    0C      3C         0421          MOVZWL  #SS$_ACCVIO,R0          ; Assume access violation
                                  0424          IFWRT   R2,(R1),10$             ; Buffer accessible?
            FF8C       31         042A  7$:     BRW     ABORT                   ; Branch if not
                                  042D  10$:
   08 59    0E         E1         042D          BBC     #IO$V_BRDCST,R9,15$     ; Branch if not brdcst bit request
   61     00A8 C5      7D         0431          MOVQ    UCB$Q_TL_BRKTHRU(R5),(R1) ; read bits (no remoting of this?)
            FF86       31         0436          BRW     FDT_FINISHIOC_OK        ; Complete I/O
```

```
                            0439  15$:
           51    6C   D0    0439              MOVL      P1(AP),R1                   ; Get address of characteristics buffer
                 9B   10    043C              BSBB      RTT_CHARSIZE                ; Size of chars buffer
           50    0C   3C    043E              MOVZWL    #SS$_ACCVIO,R0              ; Assume access violation
                            0441              IFNOWRT   R2,(R1),7$                  ; Buffer accessible?
         00D5 C5      95    0447              TSTB      UCB$B_RTT_PROECO(R5)        ; Previous version
                 12   12    044B              BNEQ      20$                         ; Nope
                 3F   AB    044D              BICW3     #IRP$M_FCODE,-              ; Obtain the modifiers
      50    20   A3         044F                        IRP$W_FUNC(R3), R0          ; to look for bad ones
    0040 8F      50   B1    0452              CMPW      R0, #IO$M_TYPEAHDCNT        ; Only good one
                 06   13    0457              BEQL      20$                         ; Ok
         0699 8F      B0    0459              MOVW      #SS$_INCOMPAT,-             ; Return quiet error
            40   A3         045D                        IRP$Q_RTT_COMPAT(R3)        ; to signal the incompatibility
      38 A3      51   D0    045F  20$:        MOVL      R1,IRP$L_MEDIA(R3)          ; Save address in packet
      32 A3      52   B0    0463              MOVW      R2,IRP$W_BCNT(R3)           ; Set size in packet
      2A A3      02   A8    0467              BISW      #IRP$M_FUNC,IRP$W_STS(R3)   ; Set READ type function
           51    18   D0    046B              MOVL      #RBF$K_HEADERLEN,R1         ; Set size of message buffer
              000D    30    046E              BSBW      ALLOC_MESSAGE               ; Allocate the message buffer
      52    18 A2     9E    0471              MOVAB     RBF$L_PARAM1(R2),R2         ; R2 points to end of data
              0303    31    0475              BRW       RTT_NETMSGSENDX             ; Send the message and exit service
```

```
                      0478          .SBTTL  ALLOC_MESSAGE, Allocate a message buffer
                      0478  ;++
                      0478  ; ALLOC_MESSAGE, Allocate a message buffer to send to remote process
                      0478  ; SET_MSGHDR,    Setup a message header for broadcast
                      0478  ;
                      0478  ; Functional description:
                      0478  ;
                      0478  ;       This routine checks that the process has sufficient buffered I/O
                      0478  ;       byte count quota for the message buffer, and then allocates the
                      0478  ;       buffer from non-paged pool.  The process's buffered I/O byte count
                      0478  ;       quota is decreased by the size of the allocated buffer and the
                      0478  ;       message header information is stored.
                      0478  ;
                      0478  ; Inputs:
                      0478  ;
                      0478  ;       R1 = size of message required
                      0478  ;       R3 = address of IRP
                      0478  ;       R4 = address of PCB
                      0478  ;
                      0478  ; Outputs:
                      0478  ;
                      0478  ;       R1 = size of buffer
                      0478  ;       R2 = address of buffer
                      0478  ;
                      0478  ;       IRP$L_SVAPTE(R3) = address of buffer
                      0478  ;       IRP$W_BOFF(R3) = size of buffer
                      0478  ;
                      0478  ;       RBF$B_TYPE(R2)  = Block type
                      0478  ;       RBF$W_SIZE(R2) = size of message buffer
                      0478  ;       RBF$W_OPCODE(R2) = I/O function
                      0478  ;       RBF$W_MOD(R2) = I/O function modifiers
                      0478  ;       RBF$L_REFID(R2) = Reference id of function
                      0478  ;       RBF$W_UNIT(R2) = Set to SVPN of the ucb (?? not used really)
                      0478  ;
                      0478  ;       If process does not have sufficient quota, the I/O request
                      0478  ;       is aborted.
                      0478  ;--
                      0478  ALLOC_ABORT:
             53 8ED0  0478          POPL    R3                      ; Restore IRP
         FF3B    31   047B          BRW     ABORT                   ; and abort the I/O
                      047E
                      047E  ALLOC_MESSAGE:                          ; Allocate message buffer
             53   DD  047E          PUSHL   R3                      ; Save packet address
   00000000'GF    16  0480          JSB     G^EXE$BUFFRQUOTA        ; Check quota
         EF 50    E9  0486          BLBC    R0,ALLOC_ABORT          ; Branch if error
                      0489  ;
                      0489  ; Allocate the message buffer
                      0489  ;
   00000000'GF    16  0489          JSB     G^EXE$ALLOCBUF          ; Allocate the buffer
         E6 50    E9  048F          BLBC    R0,ALLOC_ABORT          ; Branch if error
             53 8ED0  0492          POPL    R3                      ; Restore packet address
                      0495  ;
                      0495  ; Adjust process's quota
                      0495  ;
  50    0080 C4   D0  0495          MOVL    PCB$L_JIB(R4),R0        ; Get Job Information Block address
    20 A0    51   C2  049A          SUBL    R1,JIB$L_BYTCNT(R0)     ; Adjust buffered I/O byte count quota
    30 A3    51   B0  049E          MOVW    R1,IRP$W_BOFF(R3)       ; Save buffer size as quota
```

```
                                 04A2  ;
                                 04A2  ; Store message header information
                                 04A2  ;
                                 04A2
                                 04A2  ;
                                 04A2  ;       R0      = Clobbered
                                 04A2  ;       R1      = Buffer size
                                 04A2  ;       R2      = Buffer address
                                 04A2  ;       R3      = IRP address
                                 04A2  ;
                                 04A2
                                 04A2 SET_MSGHDR:
                                 04A2
        12 A2    50 A3     D0    04A2          MOVL    IRP$L_SEQNUM(R3), -     ; Sequence number of the operation
                                 04A7                  RBF$L_REFID(R2)
           50    1C A3     D0    04A7          MOVL    IRP$L_UCB(R3), R0       ; Unit control block address
        16 A2    74 A0     B0    04AB          MOVW    UCB$L_SVPN(R0), -       ; Bogus unit number, not used
                                 04B0                  RBF$W_UNIT(R2)
                      00   EF    04B0          EXTZV   #IRP$V_FCODE,-          ; Set requested function code
  0E A2    20 A3      06         04B2                  #IRP$S_FCODE,IRP$W_FUNC(R3),RBF$W_OPCODE(R2)
  10 A2    20 A3      3F   AB    04B7          BICW3   #IRP$M_FCODE,IRP$W_FUNC(R3),RBF$W_MOD(R2) ; Set requested modifiers
                                 04BD
                                 04BD  ;
                                 04BD  ; Setup a message header but don't depend on the irp address
                                 04BD  ; except for svapte.
                                 04BD  ;
                                 04BD
                                 04BD SET_MSGHDRX:
                                 04BD
           2C A3      52   D0    04BD          MOVL    R2,IRP$L_SVAPTE(R3)     ; Save buffer address in packet
           08 A2      51   B0    04C1          MOVW    R1,RBF$W_SIZE(R2)       ; Save buffer size in message
                      13   90    04C5          MOVB    #DYN$C_BUFIO,-          ; Set block type
                 0A A2           04C7                  RBF$B_TYPE(R2)
                 0E A2     9E    04C9          MOVAB   RBF$W_OPCODE(R2),-      ; Set address of data
                      62         04CC                  RBF$L_MSGDAT(R2)
                 04 A2     D4    04CD          CLRL    RBF$L_USRBFR(R2)        ; Set user buffer address
                      05         04D0          RSB                             ;
```

```
                        04D1          .SBTTL  RTT_INTERRUPT In.errupt handler
                        04D1  ;++
                        04D1  ; RTT_INTERRUPT, I/O completion interrupt handler
                        04D1  ;
                        04D1  ; Functional description:
                        04D1  ;
                        04D1  ;       This routine handles an I/O completion "interrupt" from the ACP.
                        04D1  ;       The I/O status and data is obtained from the response packet from
                        04D1  ;       the remote terminal handler process, and the I/O request is completed.
                        04D1  ;
                        04D1  ; Inputs:
                        04D1  ;
                        04D1  ;       R3 = address of the IRP
                        04D1  ;       R5 = address of UCB
                        04D1  ;       IRP$L_SVAPTE(R3) = address of response message
                        04D1  ;
                        04D1  ;       IPL = 0
                        04D1  ;
                        04D1  ; Outputs:
                        04D1  ;
                        04D1  ;       I/O status copied to IRP$L_IOST and I/O request posted.
                        04D1  ;
                        04D1  ;       This routine only needs to preserve R11.
                        04D1  ;--
                        04D1  RTT_INTERRUPT:                          ; I/O completion interrupt handler
   52    2C A3     D0   04D1          MOVL    IRP$L_SVAPTE(R3),R2     ; Get address of message
      51    62     D0   04D5          MOVL    (R2),R1                 ; Address of data in buffer
            01     E1   04D8          BBC     #IRP$V_FUNC,-           ; If clr not READ/SENSE/BROADCAST
     47 2A A3           04DA                  IRP$W_STS(R3),POST
            00     EF   04DD          EXTZV   #IRP$V_FCODE,-          ; Get original function code
            06          04DF                  #IRP$S_FCODE,-
   50    20 A3          04E0                  IRP$W_FUNC(R3),R0
            60     13   04E3          BEQL    POST_BROADCAST          ; If eql BROADCAST function
      27    50     91   04E5          CMPB    R0,#IO$_SENSEMODE       ; SENSEMODE function?
            1C     13   04E8          BEQL    POST_SENSE              ; If eql yes
      1B    50     91   04EA          CMPB    R0,#IO$_SENSECHAR       ; SENSECHAR function?
            17     13   04ED          BEQL    POST_SENSE              ;If eql yes - else read function
                        04EF  ;
                        04EF  ; Set up buffer to post READ
                        04EF  ;
   62    14 A1     9E   04EF          MOVAB   RDP$T_TT_RDATA+2(R1),(R2) ; Set address of data
04 A2    38 A3     D0   04F3          MOVL    IRP$L_MEDIA(R3),4(R2)   ; Set address of user buffer
         12 A1     B1   04F8          CMPW    RDP$T_TT_RDATA(R1),-    ; Size of data greater than user buffer?
         32 A3          04FB                  IRP$W_BCNT(R3)
            25     1A   04FD          BGTRU   POST                    ; If gtru yes - leave user's size
         12 A1     B0   04FF          MOVW    RDP$T_TT_RDATA(R1),-    ; Else, set size to actual data size
         32 A3          0502                  IRP$W_BCNT(R3)          ;
            1E     11   0504          BRB     POST                    ;
                        0506  ;
                        0506  ; Set up buffer to post SENSEMODE/CHAR
                        0506  ;
                        0506  POST_SENSE:                             ;
                        0506  ;
                        0506  ;       Note that for the latest protocol, either 8 or 12 bytes will come
                        0506  ;       from this part of the message.  Size is already in IRP.
                        0506  ;
   62    12 A1     9E   0506          MOVAB   RDP$Q_TT_SCHAR(R1),(R2) ; Set address of data
```

```
     04 A2    38 A3   D0  050A          MOVL    IRP$L_MEDIA(R3),4(R2)      ; Set address of user data
            00D5 C5   95  050F          TSTB    UCB$B_RTT_PROECO(R5)       ; Latest version
                 05   12  0513          BNEQ    10$                        ; Yes
              4B A5   D0  0515          MOVL    UCB$L_DEVDEPND2(R5),-      ; Return additional characters if
              1A A1       0518                  RDP$L_TT_SCHAR2(R1)        ; they are requested
                          051A     10$:
            FFC0 8F   B3  051A          BITW    #^C<IRP$M_FCODE>,-         ; Check for spawn bits only if no
              20 A3       051E                  IRP$W_FUNC(R3)             ; modifier on the sensemode
                 02   12  0520          BNEQ    20$                        ; We have modifiers
                 26   10  0522          BSBB    SENSE_SPAWN                ; Set the three bits for spawn
                          0524     20$:
                          0524     POST:                                   ; Post the I/O
     2A A3  0200 8F   A8  0524          BISW    #IRP$M_TERMIO,IRP$W_STS(R3) ; Set terminal I/O completion
              0A A1   7D  052A          MOVQ    RDP$Q_STATUS(R1),-         ; Set I/O status
              38 A3       052D                  IRP$L_IOST1(R3)            ;
              38 A3   B1  052F          CMPW    IRP$L_IOST1(R3),-          ; If normal return
                 01       0532                  #SS$_NORMAL                ;
                 0A   12  0533          BNEQ    10$                        ; Nope
              40 A3   B5  0535          TSTW    IRP$W_RTT_COMPAT(R3)       ; Check for compatibility error
                 05   13  0538          BEQL    10$                        ; Nope
              40 A3   B0  053A          MOVW    IRP$W_RTT_COMPAT(R3),-     ; Return compatibility error
              38 A3       053D                  IRP$L_IOST1(R3)            ; to user
        00000000'GF   17  053F     10$: JMP     G^COM$POST                 ; Post the I/O
                          0545     ;
                          0545     ; Post a BROADCAST completion
                          0545     ;
                          0545     POST_BROADCAST:                         ;
                          0545          BUG_CHECK BRDMSGLOST               ; NOT supposed to get here...
                      05  0549          RSB
```

```
                          054A          .SBTTL  SENSE_SPAWN  Sense for spawn
                          054A
                          054A  ;       Sense special characteristics bits for DCL spawn command.
                          054A  ;       Return bits for ctrl/c ast, outofband ast and associated mailbox.
                          054A  ;       These bits may be reused later and are not for customer application
                          054A  ;       consumption.
                          054A  ;
                          054A  ; inputs:
                          054A  ;       r1 -> RDP message
                          054A  ;
                          054A
                          054A SENSE_SPAWN:
      50     1A A1     9E 054A          MOVAB   RDP$L_TT_SCHAR2(R1), R0 ; Address of the characteristics
   60      0200 8F     AA 054E          BICW    #TT2$M_DCL_MAILBX,(R0)  ; Reset
             60 A5     D5 0553          TSTL    UCB$L_AMB(R5)           ; Any associated mailbox?
                05     13 0556          BEQL    10$                     ; No
   60      0200 8F     A8 0558          BISW    #TT2$M_DCL_MAILBX,(R0)  ; Yes, so set characteristic
                          055D 10$:
                       05 055D          RSB
```

```
                                   055E          .SBTTL  RTT_CANCEL, Cancel I/O routine
                                   055E   ;++
                                   055E   ; RTT_CANCEL, Cancels an I/O operation in progress
                                   055E   ;
                                   055E   ; Functional description:
                                   055E   ;
                                   055E   ;       This routine cancels any CTRL/C or CTRL/Y AST's that were
                                   055E   ;       requested by the cancelling process on the cancelling channel.
                                   055E   ;
                                   055E   ;       If there are no more references remaining to the device, the UCB
                                   055E   ;       is queued to the ACP to notify it that the device is no longer in
                                   055E   ;       use.  The ACP will then check that the reference count is still zero
                                   055E   ;       and remove the UCB from I/O database and deallocate it.
                                   055E   ;
                                   055E   ; Inputs:
                                   055E   ;
                                   055E   ;       R2 = negated value of the channel index number
                                   055E   ;       R3 = address of the current IRP (I/O request packet)
                                   055E   ;       R4 = address of the PCB (process control block) for the
                                   055E   ;               process canceling I/O
                                   055E   ;       R5 = address of the UCB (unit control block)
                                   055E   ;
                                   055E   ;       IPL = driver fork IPL
                                   055E   ;
                                   055E   ; Outputs:
                                   055E   ;
                                   055E   ;       DEV$M_DMT is set in UCB$L_DEVCHAR to prevent a race if someone
                                   055E   ;       assigns and deassigns another channel to the UCB before the ACP
                                   055E   ;       dequeues the UCB.
                                   055E   ;
                                   055E   ;       The routine preserves all registers except R0-R3.
                                   055E   ;--
                                   055E   .ENABLE LOCAL_BLOCK
                                   055E
                                   055E   ASSUME  CAN$C_CANCEL EQ 0
                                   055E   ASSUME  CAN$C_DASSGN EQ 1
                                   055E
            00A4          31       055E   10$:    BRW     50$
            009E          31       0561   20$:    BRW     40$
                                   0564
                                   0564   RTT_CANCEL:                                 ; Cancel an I/O operation
         00F0 8F          BB       0564           PUSHR   #^M<R4,R5,R6,R7>            ; Save registers
              04          E1       0568           BBC     #UCB$V_ONLINE,-             ; If clr unit offline - probably template
         F1 64 A5                  056A                   UCB$W_STS(R5),10$           ;
            5C A5         B5       056D           TSTW    UCB$W_REFC(R5)              ; Any more references to device?
              EF          13       0570           BEQL    20$                         ; Nope all done.
                                   0572
       56     52          D0       0572           MOVL    R2,R6                       ; Make a copy of channel number
              58          D5       0575           TSTL    R8                          ; Cancel or deassign
              0B          13       0577           BEQL    25$                         ; Cancel
                                   0579
  57    0090 C5           DE       0579           MOVAL   UCB$L_RTT_CTRLY(R5),R7      ; Get address of CTRL/Y AST list
   00000000'GF            16       057E           JSB     G^COM$FLUSHATTNS            ; Flush all cancelled AST's
                                   0584
  57    0094 C5           DE       0584   25$:    MOVAL   UCB$L_RTT_CTRLC(R5),R7      ; Get address of CTRL/C AST list
   00000000'GF            16       0589           JSB     G^COM$FLUSHATTNS            ; Flush any cancelled AST's
  57    00C4 C5           9E       058F           MOVAB   UCB$L_RTT_BANDINCL(R5), R7 ; Flush any outofband asts
```

```
      52   00C8 C5     9E  0594          MOVAB   UCB$L_RTT_BANDINMSK(R5), R2 ; mask address
         00000000'GF   16  0599          JSB     G^COM$FLUSHCTRLS         ; Flush them by channel etc
      57   009C C5     9E  059F          MOVAB   UCB$L_RTT_BANDEXCL(R5), R7 ; Flush any outofband asts
      52   0098 C5     9E  05A4          MOVAB   UCB$L_RTT_BANDEXMSK(R5), R2 ; mask address
         00000000'GF   16  05A9          JSB     G^COM$FLUSHCTRLS         ; Flush them by channel etc
                           05AF  ;
                           05AF  ; If we are talking to new version, tell him the new masks.
                           05AF  ;
              00D5 C5  95  05AF          TSTB    UCB$B_RTT_PROECO(R5)     ; Nonzero for latest
                   48  13  05B3          BEQL    30$                      ; Old version
                   22  D0  05B5          MOVL    #RBF$B_TT_OUTBAND+1+4+1+4,- ; Size of the outband message
                   51      05B7                  R1                       ; buffer
                   53  DD  05B8          PUSHL   R3                       ; Save across dirty routine
         00000000'GF   16  05BA          JSB     G^EXE$ALONONPAGED        ; Get me some memory
                   53 8ED0 05C0          POPL    R3                       ; restore packet address
                37 50  E9  05C3          BLBC    R0, 30$                  ; Hang it up for lack of space?
                           05C6
                           05C6  ;
                           05C6  ; Here comes an incredible hack.  We are going to build a message to be
                           05C6  ; transmitted which has no irp context.  It will have a REFID of zero.
                           05C6  ; To do this we need an irp address with a svapte field to save the
                           05C6  ; packet address.  We make an "irp" by passing the address of a cell in
                           05C6  ; the ucb which can be used.  The address is backed up by the svapte offset
                           05C6  ; so that for this purpose it looks like an irp.
                           05C6  ;
                           05C6
                   53  DD  05C6          PUSHL   R3                       ; Save the bad r3
      53     4C A5     9E  05C8          MOVAB   <UCB$L_SVAPTE - -        ; Make a bogus irp address
                           05CC                  IRP$L_SVAPTE>(R5), R3    ; with only a good svapte
                 FEEE  30  05CC          BSBW    SET_MSGHDRX              ; Set up the message header
                12 A2  D4  05CF          CLRL    RBF$L_REFID(R2)          ; Ref id is zero
                16 A2  B4  05D2          CLRW    RBF$W_UNIT(R2)           ; No unit specified
                   14  B0  05D5          MOVW    #RDP$B_TT_OUTBAND+1+4+1+4,- ; Size of data to be sent
                0C A2      05D7                  RBF$W_DATSIZE(R2)        ; to the server
         0E A2     23  B0  05D9          MOVW    #IO$_SETMODE, -          ; Set the op
                           05DD                  RBF$Q_OPCODE(R2)         ; code of the message
   10 A2     0400 8F   B0  05DD          MOVW    #IO$M_OUTBAND, -         ; and the modifier
                           05E3                  RBF$W_MOD(R2)            ; for the message
      52     18 A2     9E  05E3          MOVAB   RBF$B_TT_OUTBAND(R2), R2; Now build the message itself
           82      04  90  05E7          MOVB    #4, (R2)+                ; Count for include mask
              00C8 C5  D0  05EA          MOVL    UCB$L_RTT_BANDINMSK(R5),-; Include mask
                   82      05EE                  (R2)+                    ;
           82      04  90  05EF          MOVB    #4, (R2)+                ; Count for exclude mask
              0098 C5  D0  05F2          MOVL    UCB$L_RTT_BANDEXMSK(R5),- ; Now the exclude mask
                   82      05F6                  (R2)+                    ;
                 01A4  30  05F7          BSBW    RTT_NETCANSEND           ; Send the message to the server
                   53 8ED0 05FA          POPL    R3                       ; Restore the bogus irp address
                           05FD  30$:
                           05FD
                 02E7  30  05FD          BSBW    RTT_CANIRPS              ; Cancel outstanding IRPs
                   03  11  0600          BRB     50$
                           0602
                           0602  40$:
                           0602  ;
                           0602  ; Clean up the ucb after all references have gone
                           0602  ;
                 0118  30  0602          BSBW    RTT_ABORTIRPS            ; Flush all irps from queue
```

```
                          0605                                          ; Insert UCB in ACP queue
                          0605   50$:
          00F0 8F    BA   0605           POPR    #^M<R4,R5,R6,R7>       ; Restore registers
                     05   0609           RSB                            ; Return
                          060A
                          060A   .DISABLE LOCAL_BLOCK
```

```
                        060A          .SBTTL  RTT_UNSOLIC Unsolicited interrupt handler
                        060A  ;++
                        060A  ; RTT_UNSOLIC, Unsolicted interrupt handler
                        060A  ;
                        060A  ; Functional description:
                        060A  ;
                        060A  ;       This routine handles unsolicted attention messages from the remote
                        060A  ;       terminal handler process.  If the message is:
                        060A  ;
                        060A  ;               Unsolicited data: If device has any references, deliver message
                        060A  ;                                 to associated mailbox; if no references,
                        060A  ;                                 deliver a message to the Job Controller.
                        060A  ;
                        060A  ;               Hang-up:          Deliver any CNTRL/Y AST's, specifying hang-up;
                        060A  ;                                 deliver a hangup message to associated mailbox.
                        060A  ;
                        060A  ;               CTRL/C or CTRL/Y: Any corresponding AST's are delivered.
                        060A  ;
                        060A  ;               STARTRCV         Start the receive to the net.
                        060A  ;
                        060A  ; Inputs:
                        060A  ;
                        060A  ;       R3 = address of attention message
                        060A  ;       R5 = address of UCB
                        060A  ;
                        060A  ;       IPL = 0
                        060A  ;
                        060A  ; Outputs:
                        060A  ;
                        060A  ;       Message or AST(s) delivered and attention message block deallocated.
                        060A  ;
                        060A  ;--
                        060A  RTT_UNSOLIC:                                  ; Unsolicted interrupt handler
              53   DD   060A          PUSHL   R3                            ; Save address of mesage block
                        060C          DSBINT  UCB$B_FIPL(R5)                ; Synchronize access to UCB
        51    63   D0   0613          MOVL    (R3),R1                       ; Obtain the address of the data
                        0616          CASE    RDP$W_MOD(R1),<-              ; Case on message modifier type
                        0616                  UNSOL_DATA,-                  ;  Unsolicited data
                        0616                  HANGUP,-                      ;  Hangup
                        0616                  CTRLC,-                       ;  CNTRL/C
                        0616                  CTRLY,-                       ;  CNTRL/Y
                        0616                  STARTRCV,-                    ;  Start network receive
                        0616                  RTT_BRDCST,-                  ; Broadcast message for mailbox
                        0616                  RTT_OUTBAND,-                 ; Out of band ast character
                        0616                  >,LIMIT=#RBF$C_TT_UNSOL       ;
          0050     31   0629          BRW     UNSOLIC_EXIT                  ;
                        062C  ;
                        062C  ; Deliver unsolicited data notification
                        062C  ;
                        062C  UNSOL_DATA:                                   ; Unsolicited data
        54    01   9A   062C          MOVZBL  #MSG$_TRMUNSOLIC,R4           ; Set mailbox message type
           5C A5   B5   062F          TSTW    UCB$W_REFC(R5)                ; Any references to device?
              11   13   0632          BEQL    10$                           ; If eql no - notify Job Controller
        53 60 A5   D0   0634          MOVL    UCB$L_AMB(R5),R3              ; Get address of associated mailbox
              24   13   0638          BEQL    20$                           ; If eql none - forget it
     00000000'GF   16   063A          JSB     G^EXE$SNDEVMSG                ; Deliver notification to mailbox
           18 50   E9   0640          BLBC    R0,20$                        ; If lbc failure
```

```
                  19     11  0643              BRB     20$                             ;
                             0645   10$:
         53  00000000'GF     D0  0645          MOVL    G^TTY$GL_JOBCTLMB,R3            ; Get address of Job Controller mailbox
         0D 68 A5     00     E0  064C          BBS     #UCB$V_JOB,UCB$W_DEVSTS(R5),20$ ; Branch if notified already
            00000000'GF      16  0651          JSB     G^EXE$SNDEVMSG                  ; Deliver notification to mailbox
                  04 50      E9  0657          BLBC    R0,20$                          ; If lbc failure
               68 A5  01     A8  065A          BISW    #UCB$M_JOB,UCB$W_DEVSTS(R5) ; Set Job Controller notified
                             065E   20$:
                  1C     11  065E              BRB     UNSOLIC_EXIT                    ;
                             0660
                             0660   ;
                             0660   ; Deliver hangup notification
                             0660   ;
                             0660
                             0660   HANGUP:                                            ; Dataset hangup
                008C     30  0660              BSBW    RTT_HANGUP                      ; Do the hangup stuff
                  17     11  0663              BRB     UNSOLIC_EXIT                    ;
                             0665
                             0665   ;
                             0665   ; Start network receive
                             0665   ;
                             0665
                             0665   STARTRCV:
                0197     30  0665              BSBW    RTT_STARTNETRCV                 ; Start it out of line
                  12     11  0668              BRB     UNSOLIC_EXIT
                             066A
                             066A   ;
                             066A   ; Deliver any CNTRL/C AST's
                             066A   ;
                             066A
                             066A   CTRLC:                                             ; Deliver CNTRL/C AST's
         54     0094 C5      DE  066A          MOVAL   UCB$L_RTT_CTRLC(R5),R4          ; Get address of CNTRL/C AST list
                  05     11  066F              BRB     DELAST                          ;
                             0671
                             0671   ;
                             0671   ; Deliver any CNTRL/Y AST'S
                             0671   ;
                             0671
                             0671   CTRLY:                                             ; Deliver CNTRL/Y AST's
         54     0090 C5      DE  0671          MOVAL   UCB$L_RTT_CTRLY(R5),R4          ; Get address of CNTRL/Y AST list
                             0676   DELAST:                                            ;
            00000000'GF      16  0676          JSB     G^COM$DELATTNAST                ; Deliver the AST's
                             067C                                                      ;
                             067C   UNSOLIC_EXIT:                                      ; Exit unsolicited message handler
                             067C              ENBINT                                  ; Re-enable interrupts
                  50 8ED0        067F          POPL    R0                              ; Get address of message block
         0A A0    13         90  0682          MOVB    #DYN$C_BUFIO,IRP$B_TYPE(R0) ; Be sure buffer type is valid
            00000000'GF      16  0686          JSB     G^EXE$DEANONPAGED               ; Deallocate the message block
                         05      068C          RSB
```

```
                              068D
                              068D     ;+
                              068D     ; RTT_BRDCST
                              068D     ;
                              068D     ; Deliver broadcast message to the mailbox.
                              068D     ;
                              068D     ; The unit number and name of the device is fixed up in the packet first.
                              068D     ;
                              068D     ;-
                              068D
                              068D     RTT_BRDCST:
                              068D
        38 48 A5    04   E1   068D             BBC     #TT2$V_BRDCSTMBX, -              ; If we are allowing mailbox
                              0692                     UCB$L_DEVDEPND2(R5),10$          ; to receive the messages
                 60 A5   D5   0692             TSTL    UCB$L_AMB(R5)                    ; and we have a mailbox
                    33   13   0695             BEQL    10$                              ; Nope
          0E A1  54 A5   B0   0697             MOVW    UCB$W_UNIT(R5), -                ; Then fix the unit number
                              069C                     RDP$W_TT_BRDUNIT(R1)             ; in the message
             52  28 A5   D0   069C             MOVL    UCB$L_DDB(R5), R2                ; and get the proper name of
    50    14 A2   04  00 EF   06A0             EXTZV   #0, #4, DDB$T_NAME(R2), R0       ; this device for the message
                    50   D6   06A6             INCL    R0                               ; including the count
                              06A8
                    3F   BB   06A8             PUSHR   #^M<R0, R1, R2, R3, R4, R5>      ; Copy the new name and
10 A1   10   00   14 A2  50   2C   06AA        MOVC5   R0, DDB$T_NAME(R2), #0, -        ; clobber the remainder of the
                              06B2                     #RDP$C_TT_BRDNAME, -             ; stuff in the fixed length
                              06B2                     RDP$T_TT_BRDNAME(R1)             ; field
                    3F   BA   06B2             POPR    #^M<R0, R1, R2, R3, R4, R5>      ; restore the regs
                              06B4
                    38   BB   06B4             PUSHR   #^M<R3, R4, R5>                  ; Save a few
             53  0A A1   3C   06B6             MOVZWL  RDP$W_TT_BRDTOTSIZE(R1), R3      ; Size of the message
             54  0C A1   9E   06BA             MOVAB   RDP$W_TT_BRDMSG(R1), R4          ; Address of the message
             55  60 A5   D0   06BE             MOVL    UCB$L_AMB(R5), R5                ; Mailbox ucb address
           00000000'GF   16   06C2             JSB     G^EXE$WRTMAILBOX                 ; Write the message to it
                    38   BA   06C8             POPR    #^M<R3, R4, R5>                  ; and ignore the errors
                              06CA
                  FFAF   31   06CA     10$:    BRW     UNSOLIC_EXIT                     ; Go clean up the packet.
```

```
                                06CD
                                06CD     ;+
                                06CD     ; RTT_OUTBAND
                                06CD     ;
                                06CD     ; Deliver an out of band ast
                                06CD     ;-
                                06CD
                                06CD     RTT_OUTBAND:
        53    0A A1      9A     06CD             MOVZBL  RDP$B_TT_OUTBAND(R1), R3        ; Deliver the asts (char)
                 53      DD     06D1             PUSHL   R3                              ; Save the character
       54   00C4 C5      9E     06D3             MOVAB   UCB$L_RTT_BANDINCL(R5), R4      ; List address
         00000000'GF     16     06D8             JSB     G^COM$DELCTRLAST                ; Deliver the asts
                 53    8ED0     06DE             POPL    R3                              ; Recover the character
       54   009C C5      9E     06E1             MOVAB   UCB$L_RTT_BANDEXCL(R5), R4      ; List address
         00000000'GF     16     06E6             JSB     G^COM$DELCTRLAST                ; Deliver the asts
               FF8D      31     06EC             BRW     UNSOLIC_EXIT                    ; Thats all done
```

```
                          06EF        .SBTTL  RTT_HANGUP  - Perform hangup functions
                          06EF        .SBTTL  RTT_ABORTIRPS - Abort irps outstanding
                          06EF  ;++
                          06EF  ; RTT_HANGUP  Perform hangup functions
                          06EF  ; RTT_ABORTIRPS
                          06EF  ;
                          06EF  ; Functional description:
                          06EF  ;
                          06EF  ;       Deliver any CNTRL/Y AST's, specifying hang-up;
                          06EF  ;       deliver a hangup message to associated mailbox.
                          06EF  ;       Post any irps outstanding with abort.
                          06EF  ;       Set hangup status in device status.
                          06EF  ;       The ucb is passed on to the acp if there are no more
                          06EF  ;       channels open to it.
                          06EF  ;       HANGUP is called by net device errors and hangup operations
                          06EF  ;       from the line on the other end.
                          06EF  ;       ABORTIRPS is called on net device cancels and channel deassigns.
                          06EF  ;
                          06EF  ; Inputs:
                          06EF  ;
                          06EF  ;       R5 = address of UCB
                          06EF  ;
                          06EF  ;
                          06EF  ; Outputs:
                          06EF  ;
                          06EF  ;       Message or AST(s) delivered.
                          06EF  ;
                          06EF  ;--
                          06EF  RTT_HANGUP:
54    0090 C5     DE      06EF          MOVAL   UCB$L_RTT_CTRLY(R5),R4  ; Get address of CTRL/Y AST list
       50  54     D0      06F4          MOVL    R4,R0                   ; Copy list address
                          06F7  10$:
       50  60     D0      06F7          MOVL    (R0),R0                 ; Get address of next entry
           08     13      06FA          BEQL    20$                     ; If eql none
     02CC  8F     3C      06FC          MOVZWL  #SS$_HANGUP,-           ; Insert new parameter for AST
        1C A0             0700                  ACB$L_KAST+4(R0)        ;
           F3     11      0702          BRB     10$                     ;
                          0704  20$:
  00000000'GF     16      0704          JSB     G^COM$DELATTNAST        ; Deliver the AST's
       54  06     D0      070A          MOVL    #MSG$_TRMHANGUP,R4      ; Set mailbox message type
  53   60 A5      D0      070D          MOVL    UCB$L_AMB(R5),R3        ; Get associated mailbox address
           06     13      0711          BEQL    30$                     ; If eql none - forget it
  00000000'GF     16      0713          JSB     G^EXE$SNDEVMSG          ; Deliver notification to mailbox
                          0719  30$:
           08     AB      0719          BISW    #UCB$M_TT_HANGUP,-      ; Save hangup status
        68 A5             071B                  UCB$W_DEVSTS(R5)        ;
                          071D
                          071D  ;
                          071D  ;       Clean up the outstanding iirp read to network so it completes
                          071D  ;       without calling driver again.  Post all outstanding irps with
                          071D  ;       abort.
                          071D  ;
                          071D
                          071D  RTT_ABORTIRPS:
                          071D
                          071D  ;
                          071D  ;       We must be at ipl 7 or above here
```

```
RTT
Sym

IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
IRP
JIB
MAS
MAS
MSG
MSG
ORB
ORB
ORB
ORB
P1
P2
P3
P4
P5
P6
PCB
PCB
POS
POS
POS
PR$
PRM
PRM
RBF
RBF
RBF
RBF
RBF
RBF
RBF
RBF
RBF
RBF
RBF
RBF
RBF
```

```
                           071D       ;
                           071D              DSBINT  UCB$B_FIPL(R5)           ; Synchronize owth other entries
                           0724
                           0724       ;
                           0724       ;      Fix the interlock with the receive iirp so it will be deallocated
                           0724       ;      when it completes.  We must say we did so here.  The condition is
                           0724       ;      NETIRP = 1 and IRP$L_AST = 0 means that its gone.  If NETIRP = 0
                           0724       ;      it has never been allocated and given to netdriver.
                           0724       ;
                           0724
   50    00C0 C5      D0   0724              MOVL    UCB$L_RTT_NETIRP(R5),R0  ; Look at address of receive iirp
                06    13   0729              BEQL    10$                      ; Nope not here
             03 50    E8   072B              BLBS    R0,10$                   ; Dummy, all done?
             10 A0    D4   072E              CLRL    IRP$L_AST(R0)            ; Nope so tell receive iirp
   00C0 C5      01    D0   0731       10$:   MOVL    #1,UCB$L_RTT_NETIRP(R5)  ; Clobber address here
                           0736
                           0736       ;
                           0736       ;      Now we abort all of the irps that we have at this time.
                           0736       ;
                           0736
   53    00B8 D5      0F   0736       20$:   REMQUE  @UCB$L_RTT_IRPFL(R5), R3; Obtain an irp from queue
                0F    1D   073B              BVS     30$                      ; No more
      38 A3     2C    3C   073D              MOVZWL  #SS$_ABORT, -            ; Complete with abort status
                           0741                      IRP$L_IOST1(R3)          ;
             3C A3    D4   0741              CLRL    IRP$L_IOST2(R3)          ;
    00000000'GF       16   0744              JSB     G^COM$POST               ; and poast
                EA    11   074A              BRB     20$                      ; and back for more irps
                           074C
                           074C       ;
                           074C       ;      If there are no more channels to this device, then pass it on
                           074C       ;      to the acp for disposal.
                           074C       ;
                           074C
             5C A5    B5   074C       30$:   TSTW    UCB$W_REFC(R5)           ; Any channels to device?
                26    12   074F              BNEQ    50$                      ; Yes
                           0751
                01    AA   0751              BICW    #UCB$M_JOB,-             ; Clear Job Controller notified
             68 A5         0753                      UCB$W_DEVSTS(R5)         ;
                15    E2   0755              BBSS    #DEV$V_DMT,-             ; If set, UCB already queued
          1D 38 A5         0757                      UCB$L_DEVCHAR(R5),50$    ;
             53 55    D0   075A              MOVL    R5,R3                    ; Set up ucb as the packet
          52 34 A5    D0   075D              MOVL    UCB$L_VCB(R5),R2         ; Get address of VCB
          52 10 A2    D0   0761              MOVL    VCB$L_AQB(R2),R2         ; Get address of ACP AQB
    00000000'GF       16   0765              JSB     G^EXE$INSERTIRP          ; Insert UCB in ACP queue
                0A    12   076B              BNEQ    40$                      ; If neg, not first entry in queue
      51  0C A2       D0   076D              MOVL    AQB$L_ACPPID(R2),R1      ; Get ACP process ID
    00000000'GF       16   0771              JSB     G^SCH$WAKE               ; Wake the ACP process
                           0777       40$:
                           0777       50$:   ENBINT                           ; Restore IPL
                05         077A              RSB                              ;
```

```
                              077B          .SBTTL RTT_NETMSGSEND  - Send message to net driver
                              077B  ;
                              077B  ; RTT_NETMSGSENDX  -  Send message to netdriver and exit qio
                              077B  ; RTT_NETMSGSEND   -  Send message to netdriver
                              077B  ; RTT_NETCANSEND   -  Send message for cancel
                              077B  ; RTT_NETQUEPKT    -  Queue message to net driver
                              077B  ;
                              077B  ; inputs:
                              077B  ;         r2 -> address beyond message data (NETMSGSEND)
                              077B  ;         r3 -> rtt irp
                              077B  ;         r4 -> pcb
                              077B  ;         r5 -> rtt ucb
                              077B  ;
                              077B
                              077B RTT_NETMSGSENDX:
               06     10      077B          BSBB    RTT_NETMSGSEND          ; Send the message and
        00000000'GF   17      077D          JMP     G^EXE$QIORETURN         ; Return from the qio
                              0783
                              0783 RTT_NETMSGSEND:
     50   2C  A3      D0      0783          MOVL    IRP$L_SVAPTE(R3),R0     ; The buffer address
               08     13      0787          BEQL    10$                     ; none
   51   52    60      C3      0789          SUBL3   (R0),R2,R1              ; Make the length of the data
      0C  A0  51      B0      078D          MOVW    R1,RBF$W_DATSIZE(R0)    ; save in the buffer
          00C0  C5    E8      0791 10$:     BLBS    UCB$L_RTT_NETIRP(R5),-  ; We do not have a receive posted
                3A            0795                  RTT_NETHUNGUP           ; so this cannot work. We have hungup.
   00BC  D5   63      0E      0796          INSQUE  (R3), -                 ; Queue the irp on the ucb
                              079B                  @UCB$L_RTT_IRPBL(R5)
          3C  A3      D4      079B          CLRL    IRP$L_IOST2(R3)         ; No cancel has been sent yet
                              079E
                              079E RTT_NETCANSEND:                          ; Send cancel message
                              079E
             019C     30      079E          BSBW    RTT_MAKEIIRP            ; Make iirp for this message
            55 50     E9      07A1          BLBC    R0,RTT_CLEANUP          ; No memory, hangup and goaway
          08D5'CF     9E      07A4          MOVAB   W^RTT_NETWRTDONE,-      ; Place to post io
            0C A2             07A8                  IRP$L_PID(R2)           ;
  2C  A2  2C  A3      D0      07AA          MOVL    IRP$L_SVAPTE(R3), -     ; Move buffer to iirp
                              07AF                  IRP$L_SVAPTE(R2)        ;
          2C  A3      D4      07AF          CLRL    IRP$L_SVAPTE(R3)        ; drop it from rtt irp
     51   2C  A2      D0      07B2          MOVL    IRP$L_SVAPTE(R2),R1     ; fix the byte count in the iirp
  32  A2  0C  A1      B0      07B6          MOVW    RBF$W_DATSIZE(R1), -    ; from the size in the buffer
                              07BB                  IRP$W_BCNT(R2)          ;
                              07BB
                              07BB RTT_NETQUEPKT:                           ; Queue a packet to the netdriver
                              07BB
                              07BB  ;
                              07BB  ;         r2 -> net iirp
                              07BB  ;         r3 -> rtt irp
                              07BB  ;         r5 -> rtt ucb
                              07BB  ;
                              07BB
               38     BB      07BB          PUSHR   #^M<R3,R4,R5>           ; Save the magic three
         53   52      D0      07BD          MOVL    R2,R3                   ; Point to iirp
     55   1C  A3      D0      07C0          MOVL    IRP$L_UCB(R3),R5        ; The netucb from this packet
        00000000'GF   16      07C4          JSB     G^EXE$ALTQUEPKT         ; Queue iirp to netdriver
               38     BA      07CA          POPR    #^M<R3,R4,R5>           ; restore magic three
         50   01      D0      07CC          MOVL    #1,R0                   ; return success
                      05      07CF          RSB
```

```
                             07D0  1757
```

```
                                  07D0
                                  07D0  ;
                                  07D0  ;         R5 -> RTT UCB
                                  07D0  ;         R3 -> RTT IRP
                                  07D0  ;
                                  07D0  ;
                                  07D0  ; The net connection is broken, so we must post the irps that come
                                  07D0  ; in with an error code.
                                  07D0  ;
                                  07D0
                                  07D0  RTT_NETHUNGUP:
              50    2C A3    D0   07D0          MOVL    IRP$L_SVAPTE(R3),R0     ; Do we have a buffer
                       0E    13   07D4          BEQL    10$                     ; Nope
                       53    DD   07D6          PUSHL   R3                      ; Push address we care about
                    2C A3    D4   07D8          CLRL    IRP$L_SVAPTE(R3)        ; Forget we had buffer
              00000000'GF    16   07DB          JSB     G^EXE$DEANONPAGED       ; Get rid of the buffer
                       53  8ED0   07E1          POPL    R3                      ; Restore irp address
   00000000 000020E4 8F    7D   07E4  10$:    MOVQ    #SS$_LINKABORT,-        ; Return a nasty error
                    38 A3         07EE                  IRP$L_IOST1(R3)         ;
              00000000'GF    16   07F0          JSB     G^COM$POST              ; Post the irp since we don't have
                       50    D4   07F6          CLRL    R0                      ; a link anymore and return error here
                             05   07F8          RSB
                                  07F9
                                  07F9
                                  07F9          .SBTTL  RTT_CLEANUP  - Hangup terminal
                                  07F9  ;
                                  07F9  ; RTT_CLEANUP
                                  07F9  ;
                                  07F9  ;         We are in deep trouble.  Hangup the terminal to run it down
                                  07F9  ;         and return failure in r0.  This is done when we cannot obtain
                                  07F9  ;         memory for an iirp or any thing else.  IPL can be anything.
                                  07F9  ;
                                  07F9  ; inputs:
                                  07F9  ;         r5 -> rtt ucb
                                  07F9  ;
                                  07F9
                                  07F9  RTT_CLEANUP:
                                  07F9
                     FEF3    30   07F9          BSBW    RTT_HANGUP              ; Post irps and attn asts
                       50    D4   07FC          CLRL    R0                      ; return failure
                             05   07FE          RSB
```

```
                           07FF              .SBTTL  RTT_STARTNETRCV  - Start receive to net driver
                           07FF       ;
                           07FF       ; RTT_STARTNETRCV
                           07FF       ;
                           07FF       ;      Start the first receive iirp to the netdriver.  We make an iirp
                           07FF       ;      and queue it to the netdriver with a read function in it.
                           07FF       ;
                           07FF       ; inputs:
                           07FF       ;      r5 -> rtt ucb
                           07FF       ;
                           07FF
                           07FF  RTT_STARTNETRCV:
                           07FF
             00C0 C5    D5 07FF              TSTL    UCB$L_RTT_NETIRP(R5)     ; Is the iirp already out?
                  2E    12 0803              BNEQ    20$                      ; Yes, then ignore it
     00DE C5    0699 8F    B0 0805              MOVW    #SS$_INCOMPAT,UCB$W_RTT_READERR(R5) ; set initial error
                012E    30 080C              BSBW    RTT_MAKEIIRP             ; Make an iirp for use
               E7 50    E9 080F              BLBC    R0, RTT_CLEANUP          ; No good, clean it all up
          00C0 C5 52    D0 0812              MOVL    R2, UCB$L_RTT_NETIRP(R5) ; Save the address of the iirp
       0C A2    0834'CF    9E 0817              MOVAB   W^RTT_NETREADDONE, -     ; Stuff the post address
                           081D                      IRP$L_PID(R2)            ;
            20 A2    21    B0 081D              MOVW    #IO$_READLBLK, -         ; Set the function
                           0821                      IRP$W_FUNC(R2)           ;
               2C A2    D4 0821              CLRL    IRP$L_SVAPTE(R2)         ; Yes we have no buffer
         00000000'GF    B0 0824              MOVW    G^IOC$GW_MAXBUF,-        ; Set the requested size
               32 A2       082A                      IRP$W_BCNT(R2)           ;
       00 2A A2    01    E2 082C              BBSS    #IRP$V_FUNC, -           ; Say this is a read function
                           0831                      IRP$W_STS(R2), 10$       ;
                  B8    10 0831  10$:        BSBB    RTT_NETQUEPKT            ; and queue the packet to the net
                        05 0833  20$:        RSB
```

```
                          0834          .SBTTL  RTT_NETREADDONE  - Post routine for net receive
                          0834  ;
                          0834  ; RTT_NETREADDONE  Post net receive
                          0834  ;
                          0834  ;       This is the post routine for receives from the netdriver.
                          0834  ;       We look at the packet and send it to the unsolic or interrupt
                          0834  ;       routine based on the type of the message.  If the type is
                          0834  ;       not recognised or we can't find the irp, we hangup the terminal.
                          0834  ;
                          0834  ;       We are going to run this code at rtt driver ipl.
                          0834  ;
                          0834  ; inputs:
                          0834  ;       r5 -> net iirp
                          0834  ;       ipl = iopost
                          0834  ;
                          0834
                          0834  RTT_NETREADDONE:
                          0834
                38    BB  0834          PUSHR   #^M<R3,R4,R5>             ; Save the magic three
                          0836          DSBINT  #RTT$K_FIPL               ; Do this work at driver ipl
          53    55    D0  083C          MOVL    R5,R3                     ; The iirp address is here
      55    10  A3    D0  083F          MOVL    IRP$L_AST(R3),R5          ; The rtt ucb?
                1C    13  0843          BEQL    10$                       ; Its gone, we are hung up
      7D 38     A3    E9  0845          BLBC    IRP$L_IOST1(R3), 60$      ; Error? if so then hang up
      52    2C  A3    D0  0849          MOVL    IRP$L_SVAPTE(R3), R2      ; The buffer address
          51    62    D0  084D          MOVL    (R2),R1                   ; Point to message
   50     61    01    A1  0850          ADDW3   #1,RDP$W_OPCODE(R1),R0    ; Look at the opcode
                14    12  0854          BNEQ    20$                       ; Its not attention packet
          2C    A3    D4  0856          CLRL    IRP$L_SVAPTE(R3)          ; Buffer not in net packet now
        53      52    D0  0859          MOVL    R2,R3                     ; Point to buffer with r3
              FDAB    30  085C          BSBW    RTT_UNSOLIC               ; Unsolicited input attention message
                39    11  085F          BRB     40$                       ; Requeue a read
                          0861
                          0861  10$:    ENBINT                            ; Restore ipl
                38    BA  0864          POPR    #^M<R3,R4,R5>             ; Restore all the regs we saved
              006C    30  0866          BSBW    RTT_NETWRTDONE            ; Dispose of the iirp and its buffer
                      05  0869          RSB                               ;
                          086A
                50    B6  086A  20$:    INCW    R0                        ; Is this an end message?
                58    12  086C          BNEQ    60$                       ; Nope, hangup the terminal
          50    62    D0  086E          MOVL    (R2),R0                   ; Point to data
      50    04  A0    D0  0871          MOVL    RDP$L_REFID(R0),R0        ; Obtain the reference id
                23    13  0875          BEQL    40$                       ; ** Ignore refids of zero to make
                          0877                                            ; ** cancel of outofband work
   54   00B8    C5    7E  0877          MOVAQ   UCB$L_RTT_IRPFL(R5),R4    ; Look through the irps for ours
          51    54    D0  087C          MOVL    R4,R1                     ; head of queue here
          54    64    D0  087F  30$:    MOVL    (R4),R4                   ; Link through chain
          51    54    D1  0882          CMPL    R4,R1                     ; end of irps?
                3F    13  0885          BEQL    60$                       ; Yes, could not find it, hangup
      50 A4     50    D1  0887          CMPL    R0,IRP$L_SEQNUM(R4)       ; Match? on ref id
                F2    12  088B          BNEQ    30$                       ; nope
          2C    A3    D4  088D          CLRL    IRP$L_SVAPTE(R3)          ; Buffer not in net iirp now
          53    64    0F  0890          REMQUE  (R4),R3                   ; Remove the rtt irp from queue
      2C A3     52    D0  0893          MOVL    R2,IRP$L_SVAPTE(R3)       ; stick buffer there
              FC37    30  0897          BSBW    RTT_INTERRUPT             ; and call interrupt routine
                          089A  40$:
                          089A  ;
```

```
                              089A  ;          16(SP)  RTNADR
                              089A  ;          12(SP)  R5 (iirp address)
                              089A  ;          8(SP)   R4
                              089A  ;          4(SP)   R3
                              089A  ;          0(SP)   SAVED IPL (iopost)
                              089A  ;
        53    0C AE      D0   089A             MOVL    12(SP),R3                ; Obtain the net iirp
        55    1C A3      D0   089E             MOVL    IRP$L_UCB(R3),R5         ; Set the net ucb address up
        50    2C A3      D0   08A2             MOVL    IRP$L_SVAPTE(R3),R0      ; dump the buffer
                 0E      13   08A6             BEQL    50$                      ; if there is one to dump
                 53      DD   08A8             PUSHL   R3                       ; Save possibly clobbered register
        00000000'GF      16   08AA             JSB     G^EXE$DEANONPAGED        ; back into swimming pool
                 53    8ED0   08B0             POPL    R3                       ; Restore register
              2C A3      D4   08B3             CLRL    IRP$L_SVAPTE(R3)         ; forget it
        00000000'GF      B0   08B6      50$:   MOVW    G^IOC$GW_MAXBUF,-        ; setup for another read from net
              32 A3           08BC                     IRP$W_BCNT(R3)           ; with requested buffer size
        00000000'GF      16   08BE             JSB     G^EXE$ALTQUEPKT          ; queue to net driver
                 09      11   08C4             BRB     70$                      ; Now we are done here
                              08C6
                              08C6  ;
                              08C6  ;       If we had on io error in the packet, then hangup the terminal
                              08C6  ;       deallocate the packet and any buffer and exit.
                              08C6  ;       If there is no rtt ucb left anymore, just deallocate the packet
                              08C6  ;       and buffer and get out.
                              08C6  ;
                              08C6
               FE26      30   08C6      60$:   BSBW    RTT_HANGUP               ; Bad error - hangup the terminal
        55    0C AE      D0   08C9             MOVL    12(SP),R5                ; Net iirp to r5
                 06      10   08CD             BSBB    RTT_NETWRTDONE           ; Dump the buffer and the iirp
                              08CF      70$:   ENBINT                           ; Restore the ipl
                 38      BA   08D2             POPR    #^M<R3,R4,R5>            ; restore registers of iopost
                         05   08D4             RSB
                         08D5             .SBTTL  RTT_NETWRTDONE  - Post routine for net write
                         08D5       ;
                         08D5       ; RTT_NETWRTDONE
                         08D5       ;
                         08D5       ;       Enter here to post writes to net also.
                         08D5       ;       Deallocate the iirp and the message if any.
                         08D5       ;
                         08D5       ;       r5 -> iirp
                         08D5       ;       ipl = iopost or higher
                         08D5       ;
                         08D5
                         08D5       RTT_NETWRTDONE:
                         08D5
   50   2C A5    D0      08D5               MOVL    IRP$L_SVAPTE(R5),R0     ; Buffer on this iirp?
           02    13      08D9               BEQL    10$                     ; nope
           03    10      08DB               BSBB    20$                     ; deallocate the buffer
      50   55    D0      08DD       10$:    MOVL    R5,R0                   ; Now for the iirp itself
  00000000'GF    16      08E0       20$:    JSB     G^EXE$DEANONPAGED       ; back to the pool
                 05      08E6               RSB
```

```
                          08E7          .SBTTL  RTT_CANIRPS  - Cancel irps
                          08E7  ;
                          08E7  ; RTT_CANIRPS
                          08E7  ;
                          08E7  ;       Cancel irps by sending a message to the terminal system.
                          08E7  ;
                          08E7  ; inputs:
                          08E7  ;       r4 -> pcb for process
                          08E7  ;       r5 -> rtt ucb
                          08E7  ;       r6 -> channel
                          08E7  ;
                          08E7
                          08E7 RTT_CANIRPS:
                          08E7
        007C 8F     BB    08E7          PUSHR   #^M<R2,R3,R4,R5,R6>
   56   00B8 C5     7E    08EB          MOVAQ   UCB$L_RTT_IRPFL(R5),R6  ; Point to the irp queue
               56   DD    08F0          PUSHL   R6                      ; save its address
                          08F2  ;
                          08F2  ;       20(SP)  R6
                          08F2  ;       16      R5
                          08F2  ;       12      R4
                          08F2  ;       8       R3
                          08F2  ;       4       R2
                          08F2  ;       0       IRP LIST HEAD
                          08F2
          56   66   D0    08F2 10$:     MOVL    (R6),R6                 ; Point to next irp
          6E   56   D1    08F5          CMPL    R6,(SP)                 ; End of queue?
               3E   13    08F8          BEQL    20$                     ; Yes
   28 A6    14 AE   B1    08FA          CMPW    20(SP),IRP$W_CHAN(R6)   ; Is this the correct channel?
               F1   12    08FF          BNEQ    10$                     ; Nope, try next?
   0C A6    60 A4   D1    0901          CMPL    PCB$L_PID(R4), -        ; Do the pids match?
                          0906                  IRP$L_PID(R6)           ;
               EA   12    0906          BNEQ    10$                     ; Nope, try next
          53   56   D0    0908          MOVL    R6,R3                   ; Set up as the irp of choice
            3C A3   D5    090B          TSTL    IRP$L_IOST2(R3)         ; Did we send a cancel?
               28   12    090E          BNEQ    20$                     ; We are done, just return
          51   18   D0    0910          MOVL    #RBF$W_UNIT+2, R1       ; Get a message buffer for cancel
               53   DD    0913          PUSHL   R3                      ; Save across call
     00000000'GF    16    0915          JSB     G^EXE$ALONONPAGED       ;
             53 8ED0      0918          POPL    R3                      ; Its clobbered if quick irps are gone
          11   50   E9    091E          BLBC    R0,15$                  ; If error, just say we did it
             FB7E   30    0921          BSBW    SET_MSGHDR              ; build the message
                          0924          ASSUME  RBF$W_MOD EQ -
                          0924                  RBF$W_OPCODE+2
               38   D0    0924          MOVL    #IO$_ACPCONTROL,-       ; The message opcode and modifier
            0E A2         0926                  RBF$Q_OPCODE(R2)        ;
               0A   B0    0928          MOVW    #RDP$Q_UNIT+2,-         ; The datasize
            0C A2         092A                  RBF$W_DATSIZE(R2)       ;
                          092C  ;       MOVL    R2,IRP$L_SVAPTE(R3)     ; Save the buffer address **
             FE6F   30    092C          BSBW    RTT_NETCANSEND          ; Send the message
          06   50   E9    092F          BLBC    R0,20$                  ; Error, IRPS are all gone
   3C A3       01   D0    0932 15$:     MOVL    #1,IRP$L_IOST2(R3)      ; Mark for we sent it
               BA   11    0936          BRB     10$                     ; try another irp
                          0938
        007E 8F     BA    0938 20$:     POPR    #^M<R1,R2,R3,R4,R5,R6>  ; Restore regs and return
                    05    093C          RSB                             ; Discard stack longword to r1
                    093D          .SBTTL RTT_MAKEIIRP  - Manufacture an internal irp
                    093D    ;
                    093D    ; RTT_MAKEIIRP
                    093D    ;
                    093D    ;       Make an internal IRP for sending to the netdriver.
                    093D    ;       If we can't get the space, return failure.
                    093D    ;
                    093D    ; inputs:
                    093D    ;       r3 -> rtt irp
                    093D    ;       r5 -> rtt ucb
                    093D    ;
                    093D    ; outputs:
                    093D    ;       r0 = success or fail
                    093D    ;
                    093D
                    093D    RTT_MAKEIIRP:
                    093D
   51    C4 8F    9A 093D          MOVZBL  #IRP$C_LENGTH,R1          ; Obtain a buffer of correct size
             53    DD 0941          PUSHL   R3                        ; Save across call to get memory
  00000000'GF      16 0943          JSB     G^EXE$ALONONPAGED         ; from dynamic memory
             53  8ED0 0949          POPL    R3                        ; Restore irp address
         3A  50    E9 094C          BLBC    R0,10$                    ; No memory left, so return error
    0A A2    0A    90 094F          MOVB    #DYN$C_IRP, -             ; Set the type and size fields
                      0953                  IRP$B_TYPE(R2)            ;
    08 A2    51    B0 0953          MOVW    R1,IRP$W_SIZE(R2)         ;
         0C  A2    D4 0957          CLRL    IRP$L_PID(R2)             ; No pid here
    10 A2    55    D0 095A          MOVL    R5,IRP$L_AST(R2)          ; Save the rtt ucb field
       00B4  C5    D0 095E          MOVL    UCB$L_RTT_NETWIND(R5),-   ; Set up the window
         18  A2       0962                  IRP$L_WIND(R2)            ;
       00B0  C5    D0 0964          MOVL    UCB$L_RTT_NETUCB(R5),-    ; and the ucb for the net
         1C  A2       0968                  IRP$L_UCB(R2)             ;
             20    B0 096A          MOVW    #IO$_WRITELBLK,-          ; the function
         20  A2       096C                  IRP$W_FUNC(R2)            ;
    23 A2    04    90 096E          MOVB    #4,IRP$B_PRI(R2)          ; priority of this in queue
             01    B0 0972          MOVW    #IRP$M_BUFIO,-            ; Its a buffered io function
         2A  A2       0974                  IRP$W_STS(R2)             ; and assume a write
         30  A2    B4 0976          CLRW    IRP$W_BOFF(R2)            ; no quota to return for iirp
         38  A2    7C 0979          CLRQ    IRP$L_IOST1(R2)           ; no status yet
                      097C          ASSUME  IRP$L_OBCNT -
                      097C                  EQ -
                      097C                  IRP$L_ABCNT+4
         40  A2    7C 097C          CLRQ    IRP$L_ABCNT(R2)           ; Some more byte counts
         50  A3    D0 097F          MOVL    IRP$L_SEQNUM(R3),-        ; Grab a quick sequence number
         50  A2       0982                  IRP$L_SEQNUM(R2)          ;
         58  A3    D0 0984          MOVL    IRP$L_ARB(R3),-           ; Access rights block, incase needed
         58  A2       0987                  IRP$L_ARB(R2)             ;
             05       0989    10$:  RSB
```

```
                                        098A  2047          .SBTTL  RTT_END, End of driver
                                        098A  2048
                                        098A  2049 ;
                                        098A  2050 ; Label that marks the end of the driver
                                        098A  2051 ;
                                        098A  2052 RTT_END:
                                        098A  2053          .END
```

```
                                                                  C  4
RTTDRIVER                         - Remote Terminal Driver                  16-SEP-1984 00:03:56  VAX/VMS Macro V04-00          Page  50
Symbol table                                                                 5-SEP-1984 00:17:28  [DRIVER.SRC]RTTDRIVER.MAR;1         (32)

$$$                             = 00000020 R      02      EXE$FINISHIOC                    *********    X   03
$$OP                            = 00000002                EXE$INSERTIRP                    *********    X   03
ABORT                             000003B9 R      03      EXE$MAXACMODE                    *********    X   03
ACB$L_KAST                      = 00000018                EXE$PROBER                       *********    X   03
ALLOC_ABORT                       00000478 R      03      EXE$QIORETURN                    *********    X   03
ALLOC_MESSAGE                     0000047E R      03      EXE$READCHK                      *********    X   03
AQB$L_ACPPID                    = 0000000C                EXE$SNDEVMSG                     *********    X   03
AT$_NULL                        = 00000005                EXE$WRITECHK                     *********    X   03
BUFADDR                         = 00000000                EXE$WRTMAILBOX                   *********    X   03
BUFSIZE                         = 00000004                FDT_FINISHIOC                     000003C2 R      03
BUG$_BRDMSGLOST                  *********    X   03      FDT_FINISHIOC_OK                  000003BF R      03
CAN$C_CANCEL                    = 00000000                FUNCTAB_LEN                     = 00000040
CAN$C_DASSGN                    = 00000001                GET_PARAMS                        000003C8 R      03
CHK_READERR                       00000276 R      03      HANGUP                            00000660 R      03
COM$DELATTNAST                   *********    X   03      INIADDR                         = 00000018
COM$DELCTRLAST                   *********    X   03      INIOFFSET                       = 00000024
COM$FLUSHATTNS                   *********    X   03      INISIZE                         = 0000001C
COM$FLUSHCTRLS                   *********    X   03      IO$M_CTRLCAST                   = 00000100
COM$POST                         *********    X   03      IO$M_CTRLYAST                   = 00000080
COM$SETATTNAST                   *********    X   03      IO$M_EXTEND                     = 00008000
COM$SETCTRLAST                   *********    X   03      IO$M_HANGUP                     = 00000200
CRB$L_INTD                      = 00000024                IO$M_OUTBAND                    = 00000400
CTRLC                             0000066A R      03      IO$M_TIMED                      = 00000080
CTRLY                             00000671 R      03      IO$M_TYPEAHDCNT                 = 00000040
CTRL_CY                           00000347 R      03      IO$V_BRDCST                     = 0000000E
DC$_TERM                        = 00000042                IO$V_BREAKTHRU                  = 00000009
DDB$L_ACPD                      = 00000010                IO$V_EXTEND                     = 0000000F
DDB$L_DDT                       = 0000000C                IO$V_INCLUDE                    = 0000000B
DDB$T_NAME                      = 00000014                IO$V_MAINT                      = 00000006
DELAST                            00000676 R      03      IO$V_RD_MODEM                   = 00000007
DEV$M_AVL                       = 00040000                IO$_ACPCONTROL                  = 00000038
DEV$M_CCL                       = 00000002                IO$_READLBLK                    = 00000021
DEV$M_IDV                       = 04000000                IO$_READPBLK                    = 0000000C
DEV$M_NNM                       = 00000200                IO$_READPROMPT                  = 00000037
DEV$M_ODV                       = 08000000                IO$_READVBLK                    = 00000031
DEV$M_REC                       = 00000001                IO$_SENSECHAR                   = 0000001B
DEV$M_RTT                       = 00000004                IO$_SENSEMODE                   = 00000027
DEV$M_TRM                       = 00000004                IO$_SETCHAR                     = 0000001A
DEV$V_DMT                       = 00000015                IO$_SETMODE                     = 00000023
DPT$C_LENGTH                    = 00000038                IO$_TTYREADALL                  = 0000003A
DPT$C_VERSION                   = 00000004                IO$_TTYREADPALL                 = 0000003B
DPT$INITAB                        00000038 R      02      IO$_VIRTUAL                     = 0000003F
DPT$REINITAB                      00000081 R      02      IO$_WRITELBLK                   = 00000020
DPT$TAB                           00000000 R      02      IO$_WRITEPBLK                   = 0000000B
DYN$C_BUFIO                     = 00000013                IO$_WRITEVBLK                   = 00000030
DYN$C_CRB                       = 00000005                IOC$GW_MAXBUF                    *********    X   03
DYN$C_DDB                       = 00000006                IOC$MNTVER                       *********    X   03
DYN$C_DPT                       = 0000001E                IOC$RETURN                       *********    X   03
DYN$C_IRP                       = 0000000A                IRP$B_PRI                       = 00000023
DYN$C_ORB                       = 00000049                IRP$B_TYPE                      = 0000000A
DYN$C_UCB                       = 00000010                IRP$C_LENGTH                    = 000000C4
EXE$ABORTIO                      *********    X   03      IRP$L_ABCNT                     = 00000040
EXE$ALLOCBUF                     *********    X   03      IRP$L_ARB                       = 00000058
EXE$ALONONPAGED                  *********    X   03      IRP$L_AST                       = 00000010
EXE$ALTQUEPKT                    *********    X   03      IRP$L_IOST1                     = 00000038
EXE$BUFFRQUOTA                   *********    X   03      IRP$L_IOST2                     = 0000003C
EXE$DEANONPAGED                  *********    X   03      IRP$L_MEDIA                     = 00000038
```

```
IRP$L_OBCNT                     = 00000044                    RBF$L_TT_TIMOUT                 = 0000001C
IRP$L_PID                       = 0000000C                    RBF$L_USRBFR                    = 00000004
IRP$L_SEQNUM                    = 00000050                    RBF$Q_TT_CHAR                   = 00000018
IRP$L_SVAPTE                    = 0000002C                    RBF$T_TT_TERM                   = 00000020
IRP$L_UCB                       = 0000001C                    RBF$T_TT_WDATA                  = 00000020
IRP$L_WIND                      = 00000018                    RBF$W_DATSIZE                   = 0000000C
IRP$M_BUFIO                     = 00000001                    RBF$W_MOD                       = 00000010
IRP$M_FCODE                     = 0000003F                    RBF$W_OPCODE                    = 0000000E
IRP$M_FUNC                      = 00000002                    RBF$W_SIZE                      = 00000008
IRP$M_TERMIO                    = 00000200                    RBF$W_UNIT                      = 00000016
IRP$Q_TT_STATE                  = 00000040                    RDP$B_TT_OUTBAND                = 0000000A
IRP$S_FCODE                     = 00000006                    RDP$C_TT_BRDNAME                = 00000010
IRP$V_FCODE                     = 00000000                    RDP$L_REFID                     = 00000004
IRP$V_FUNC                      = 00000001                    RDP$L_TT_SCHAR2                 = 0000001A
IRP$W_BCNT                      = 00000032                    RDP$Q_STATUS                    = 0000000A
IRP$W_BOFF                      = 00000030                    RDP$Q_TT_SCHAR                  = 00000012
IRP$W_CHAN                      = 00000028                    RDP$T_TT_BRDNAME                = 00000010
IRP$W_FUNC                      = 00000020                    RDP$T_TT_RDATA                  = 00000012
IRP$W_RTT_COMPAT                = 00000040                    RDP$W_MOD                       = 00000002
IRP$W_SIZE                      = 00000008                    RDP$W_OPCODE                    = 00000000
IRP$W_STS                       = 0000002A                    RDP$W_TT_BRDMSG                 = 0000000C
JIB$L_BYTCNT                    = 00000020                    RDP$W_TT_BRDTOTSIZE             = 0000000A
MASKH                           = 00000008                    RDP$W_TT_BRDUNIT                = 0000000E
MASKL                           = 04000000                    RDP$W_UNIT                      = 00000008
MSG$_TRMHANGUP                  = 00000006                    READ_ERROR                        0000019A R     03
MSG$_TRMUNSOLIC                 = 00000001                    READ_LOCAL                      = 00000028
ORB$B_FLAGS                     = 0000000B                    REM$C_CURECO                    = 00000001
ORB$L_OWNER                     = 00000000                    REM$C_CURVRS                    = 00000001
ORB$M_PROT_16                   = 00000001                    REM$C_LNK_READ                  = 00000002
ORB$W_PROT                      = 00000018                    REM$C_MAXDEVS                   = 0000000A
P1                              = 00000000                    REM$C_MAXLINKS                  = 00000010
P2                              = 00000004                    REM$C_MAXUNITS                  = 00000010
P3                              = 00000008                    REM$C_MBX_READ                  = 00000001
P4                              = 0000000C                    REM$C_ST_ATTRIB                 = 00000002
P5                              = 00000010                    REM$C_ST_CONFIG                 = 00000001
P6                              = 00000014                    RTT$DDT                           00000000 RG    03
PCB$L_JIB                       = 00000080                    RTT$K_FIPL                      = 00000008
PCB$L_PID                       = 00000060                    RTT_ABORTIRPS                     0000071D R     03
POST                              00000524 R     03           RTT_BRDCST                        0000068D R     03
POST_BROADCAST                    00000545 R     03           RTT_CANCEL                        00000564 R     03
POST_SENSE                        00000506 R     03           RTT_CANIRPS                       000008E7 R     03
PR$_IPL                         = 00000012                    RTT_CHARSIZE                      000003D9 R     03
PRMADDR                         = 00000008                    RTT_CLEANUP                       000007F9 R     03
PRMSIZE                         = 0000000C                    RTT_ECOQ                          000003F8 R     03
RBF$B_TT_OUTBAND                = 00000018                    RTT_END                           0000098A R     03
RBF$B_TYPE                      = 0000000A                    RTT_FUNCTABLE                     00000038 R     03
RBF$C_TT_UNSOL                  = 00000000                    RTT_HANGUP                        000006EF R     03
RBF$K_HEADERLEN                 = 00000018                    RTT_INTERRUPT                     000004D1 R     03
RBF$L_MSGDAT                    = 00000000                    RTT_MAKEIIRP                      0000093D R     03
RBF$L_PARAM1                    = 00000018                    RTT_NETCANSEND                    0000079E R     03
RBF$L_REFID                     = 00000012                    RTT_NETHUNGUP                     000007D0 R     03
RBF$L_TT_BCNT                   = 00000018                    RTT_NETMSGSEND                    00000783 R     03
RBF$L_TT_CARCON                 = 0000001C                    RTT_NETMSGSENDX                   0000077B R     03
RBF$L_TT_CHAR2                  = 0000002C                    RTT_NETQUEPKT                     000007BB R     03
RBF$L_TT_FILL                   = 00000024                    RTT_NETREADDONE                   00000834 R     03
RBF$L_TT_PARITY                 = 00000028                    RTT_NETWRTDONE                    000008D5 R     03
RBF$L_TT_SPEED                  = 00000020                    RTT_OUTBAND                       000006CD R     03
```

```
RTT_READ                         000000C5 R      03    UCB$L_DEVDEPEND                 = 00000044
RTT_SENSEMODE                    00000408 R      03    UCB$L_DEVDEPND2                 = 00000048
RTT_SETMODE                      00000287 R      03    UCB$L_RTT_BANDEXCL              = 0000009C
RTT_STARTNETRCV                  000007FF R      03    UCB$L_RTT_BANDEXMSK             = 00000098
RTT_UNSOLIC                      0000060A R      03    UCB$L_RTT_BANDINCL              = 000000C4
RTT_WRITE                        00000078 R      03    UCB$L_RTT_BANDINMSK             = 000000C8
RT_READ_ITMLST                   000001A0 R      03    UCB$L_RTT_CTRLC                 = 00000094
SCH$WAKE                         ********    X   03    UCB$L_RTT_CTRLY                 = 00000090
SENSE_SPAWN                      0000054A R      03    UCB$L_RTT_DEVDEPEND2            = 00000048
SET_BRDCST                       00000303 R      03    UCB$L_RTT_IRPBL                 = 000000BC
SET_CHAR                         000002C8 R      03    UCB$L_RTT_IRPFL                 = 000000B8
SET_CONNECT                      000002FB R      03    UCB$L_RTT_NETIRP                = 000000C0
SET_CTRLC                        0000033C R      03    UCB$L_RTT_NETUCB                = 000000B0
SET_CTRLY                        00000316 R      03    UCB$L_RTT_NETWIND               = 000000B4
SET_DISCONNECT                   000002FB R      03    UCB$L_SVAPTE                    = 00000078
SET_HANGUP                       0000034A R      03    UCB$L_SVPN                      = 00000074
SET_MAINT                        000002FB R      03    UCB$L_TL_BANDQUE                = 0000009C
SET_MESSAGE                      0000034A R      03    UCB$L_TL_CTLPID                 = 000000A4
SET_MSGHDR                       000004A2 R      03    UCB$L_TL_CTRLC                  = 00000094
SET_MSGHDRX                      000004BD R      03    UCB$L_TL_CTRLY                  = 00000090
SET_NOP                          00000313 R      03    UCB$L_TL_OUTBAND                = 00000098
SET_OUTBAND                      0000037B R      03    UCB$L_VCB                       = 00000034
SET_PID                          0000030D R      03    UCB$M_JOB                       = 00000001
SS$_ABORT                      = 0000002C              UCB$M_TT_HANGUP                 = 00000008
SS$_ACCVIO                     = 0000000C              UCB$Q_TL_BRKTHRU                = 000000A8
SS$_BADPARAM                   = 00000014              UCB$V_JOB                       = 00000000
SS$_DEVREQERR                  = 00000334              UCB$V_ONLINE                    = 00000004
SS$_HANGUP                     = 000002CC              UCB$V_TT_HANGUP                 = 00000003
SS$_ILLIOFUNC                  = 000000F4              UCB$W_CT_QCTPCNT                = 000000DE
SS$_INCOMPAT                   = 00000699              UCB$W_DEVBUFSIZ                 = 00000042
SS$_LINKABORT                  = 000020E4              UCB$W_DEVSTS                    = 00000068
SS$_NORMAL                     = 00000001              UCB$W_REFC                      = 0000005C
STARTRCV                         00000665 R      03    UCB$W_RTT_READERR               = 000000DE
TIMEOUT                        = 00000020              UCB$W_STS                       = 00000064
TRM$_LASTITM                   = 0000000A              UCB$W_UNIT                      = 00000054
TRMADDR                        = 00000010              UNSOLIC_EXIT                      0000067C R      03
TRMSIZE                        = 00000014              UNSOL_DATA                        0000062C R      03
TT$V_HALFDUP                   = 00000014              VCB$L_AQB                       = 00000010
TT$_UNKNOWN                    = 00000000
TT2$M_DCL_MAILBX               = 00000200
TT2$V_BRDCSTMBX                = 00000004
TTY$GL_DEFCHAR                   ********    X   02
TTY$GL_JOBCTLMB                  ********    X   03
TTY$GL_OWNUIC                    ********    X   02
TTY$GW_DEFBUF                    ********    X   02
TTY$GW_PROT                      ********    X   02
UCB$B_DEVCLASS                 = 00000040
UCB$B_DEVTYPE                  = 00000041
UCB$B_DIPL                     = 0000005E
UCB$B_FIPL                     = 0000000B
UCB$B_RTT_PROECO               = 000000D5
UCB$K_RTT_LEN                  = 00000138
UCB$K_RTT_LENGTH               = 00000138
UCB$L_AMB                      = 00000060
UCB$L_DDB                      = 00000028
UCB$L_DEVCHAR                  = 00000038
UCB$L_DEVCHAR2                 = 0000003C
```

```
                                              +----------------+
                                              ! Psect synopsis !
                                              +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           00000000  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$105_PROLOGUE                 0000008C  (  140.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$115_DRIVER                   0000098A  ( 2442.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG

                                           +------------------------+
                                           ! Performance indicators !
                                           +------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     32    00:00:00.05    00:00:01.45
Command processing                138    00:00:00.48    00:00:03.43
Pass 1                            801    00:00:25.20    00:01:30.56
Symbol table sort                   0    00:00:03.86    00:00:13.19
Pass 2                            351    00:00:05.52    00:00:21.78
Symbol table output                38    00:00:00.22    00:00:00.37
Psect synopsis output               3    00:00:00.01    00:00:00.01
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals             1365    00:00:35.34    00:02:10.80

The working set limit was 2700 pages.
211039 bytes (413 pages) of virtual memory were used to buffer the intermediate code.
There were 190 pages of symbol table space allocated to hold 3595 non-local and 92 local symbols.
2053 source lines were read in Pass 1, producing 23 object records in Pass 2.
62 pages of virtual memory were used to define 59 macros.

                                          +--------------------------+
                                          ! Macro library statistics !
                                          +--------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[SHRLIB]REM.MLB;1                        2
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                      39
_$255$DUA28:[SYSLIB]STARLET.MLB;2                   15
TOTALS (all libraries)                              56

3925 GETS were required to define 56 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RTTDRIVER/OBJ=OBJ$:RTTDRIVER MSRC$:RTTDRIVER/UPDATE=(ENH$:RTTDRIVER)+EXECML$/LIB+SHRLIB$:REM/LIB
```

0115 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PUDRIVER
LIS

RTTDRIVER
LIS

PATABLES
LIS

PASCSCTL
LIS

0116 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY